# APPLICATION OF SYMBOLICAL KINEMATICS TO REAL-TIME VEHICLE DYNAMICS

Final Technical Report

by

A. Kecskeméthy and Th. Krupp
(August 1995)

United States Army

EUROPEAN RESEARCH OFFICE OF THE U.S. ARMY

London England

CONTRACT NUMBER N68171-94-C-9091

Dr.-Ing. A. Kecskeméthy

Approved for Public Release; distribution unlimited

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | | Final Aug. 1, 1994 - Aug. 31, 1995 |

**4. TITLE AND SUBTITLE**
Application of Symbolical Kinematics to Real-Time Vehicle Dynamics

**5. FUNDING NUMBERS**
C N68171-94-C-9091
PR WK2Q6C-7349-AN01

**6. AUTHOR(S)**
Andrés Kecskeméthy
Thorsten Krupp

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Fachgebiet Mechatronik
Gerhard-Mercator-Universität - GH Duisburg
Lotharstr. 1
47058 Duisburg
Germany

**8. PERFORMING ORGANIZATION REPORT NUMBER**
FG 16-95-02

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
European Research Office - USARDSG-UK
Fiscal Office
Edison House
223 Old Marylebone Road
London NW1 5TH, United Kingdom

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**
Prepared in cooperation with
Dr. Roger Wehage
System Simulation & Tech. Division, AMSTA-RY, U.S. Army TARDEC

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
UNCLASSIFIED

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** (Maximum 200 words)

A computer-oriented integrated approach for the automatic generation of symbolical expressions for the position, velocity and acceleration problems of spatial, multiple-loop multibody systems is developed. All processing steps, from the topological analysis of the interconnection structure to the final production of executable statements in a standard programming language, such as "C", are integrated into one single single piece of code, written in Mathematica. Special subsystems, such as planar or spherical mechanism parts, subsystems featuring closed-form, i.e., analytic solutions, and subsystems which have to be solved iteratively, are recognized and processed accordingly. These tasks involve, among others, the generation of minimal cycle sets, the detection of invariant transformation groups in loops, and the recognition of recursive solution flows in multiple-loop mechanisms. All processing steps are fully operational and produce the desired expressions from a minimal input comprising the system adjacency matrix, the list of variable joint parameters, and the desired set of input variables. A SOLVAS-compatible interface insures the applicability of the package in the setting of vehicle dynamics by its integration in the libraries developed at the System Simulation & Technical Division Group at U.S. Army TARDEC. The procedures and the code are illustrated by several examples.

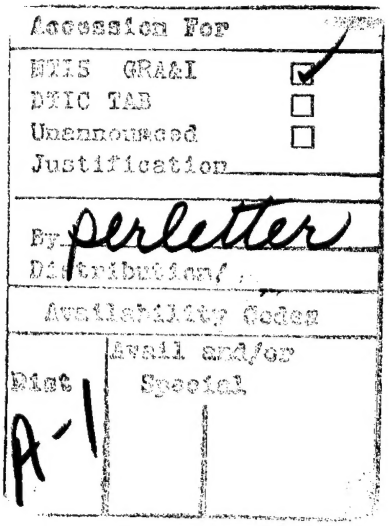
**14. SUBJECT TERMS**
Kinematics of vehicles, closed form solutions, symbolical computation, multibody systems, closed-loop analysis

**15. NUMBER OF PAGES**

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |

# Contents



# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Statement of the Problem

For the design of modern road vehicles, exact real-time computer simulation is becoming an indispensable requirement. Because of the inherent model complexity, however, such highly efficient models are only feasible with the help of fast computer equipment and/or customized model equations. The purpose of this work is to exploit the potentials of optimization of model efficiency by incorporation of closed-form solutions in the kinematics processing, and to device schemes for an automated generation of such solutions for the general spatial multiple-loop topologies that are typical in vehicle models.

## 1.2 Historical Background

The simulation of road vehicles is a subject of growing scientific interest since now almost 20 years. The objective of these efforts has been to achieve computer models that mimic the real system as exact and efficiently as possible and thus make it possible to replace experimental setups which are costly and limited in their possibilities with software components that are cheap and flexible. Hereby, the approach of multibody analysis, i. e., the analysis of the motion of a set of rigid bodies connected through ideal joints, has evolved as one of the principal investigation tools for the analysis of comfort, noise and stability properties of vehicles.

Early approaches for the computer-oriented analysis of multibody dynamics focused primarily on the broad applicability of the resulting code, using as building blocks rather simple but highly generic elements for the corresponding equations. In these approaches, the model for the overall system is expressed by a large sparse matrix resulting from the assembly of the individual element models. This model is then solved by applying appropriate numeric algorithms to it, that, in order to work properly, have to be interweaved with the model equations in a highly sophisticated manner. The described methodology emanated from the work of Orlandea, Chase and Calahan 1979 and has since then served as a paradigm for a large number of related approaches, among which are the works of Wehage and Haug 1982, Nikravesh and Haug 1982 and Garcia de Jalón et al. 1987. All these approaches have in common that they do not focus primarily on optimizing the generated equations with respect to particular geometrical properties, but try to apply general numerical methods which are also used in other application fields. This is expressed most clearly in the introduction of the paper Orlandea 1987: "When ADAMS was conceived, the basis for the formulation was to look at efficient numerical algorithms and then to formulate the general mechanical problem so that it fits the numerical methods".

Besides this so-called *numerical methods*, several further approaches were developed, mostly in Europe, in which either graph-theoretic methods (Wittenburg 1977) or symbolical processing schemes (Kreuzer 1979) are applied. These methods, termed *minimal coordinate approaches*, yield systems of pure ordinary differential equations of minimal order which are much more efficient and reliable than the equations obtained by the sparse matrix approaches. Furthermore additional gains in effectivity can be achieved by taking into consideration the particular geometric and/or dynamical properties of the system at hand and eliminating redundant operations from the outset. The minimal coordinate approach has been the basis for a number of related methods, such as the *vector network method* (Andrews and Kesavan 1975), *Kane's method* (Kane and Levinson 1985), and the method proposed by the author (Hiller et al. 1986). The advantages of the minimal coordinate approach are its high computational efficiency, the tight relationship between the computer model and the underlying mechanical system, and the simplicity and thus better comprehensibility of the underlying algorithms and intermediate results for the practising engineering.

A related substantial contribution to the evolution of efficient multibody simulation software has been to incorporate methods developed for the dynamic analysis of robotic systems (Walker and Orin 1982) into the methodology for treating general multibody systems. In particular, the so-called *order(n)* methods (Brandl, Johanni and Otter 1986), have been used as a basis for powerful, parallelizable codes (Tsai and Haug 1991a and Tsai and Haug 1991b). These contributions have given further evidence that a sound mechanical modeling helps to boost simulation performance and quality even for large complex structures such as the vehicle systems on which this research is focused.

Due to these accomplishments, the minimal coordinate approaches have gained increasing acceptance in the last decade, and have become the method of choice when highly efficient and robust code is an issue. However, a serious drawback of the minimal coordinate approach has been that the equations striven for are very difficult to obtain, so that even with the help of modern computer software the task of generating optimized expressions for the dynamical equations of systems of high complexity such as vehicles, trailers, caterpillars, etc. have remained until recently an open issue. This is a consequence of a phenomenon known as *intermediate expression swell* (Nielan and Kane 1986), which describes the rapid growth of the number and size of the intermediate terms emerging in the process of equation generation. This phenomenon is responsible for most processing algorithms falling short of reaching the end of the equation-generation algorithm due to storage-space limitations — even for subsystems of modest complexity such as a double-wishbone wheel suspension.

To tackle this problem, two alternatives have been proposed. One is to describe the nonlinear motion behaviour of the constituent parts of the system by polynomial approximation. Here, a complex system is dissected into simpler components, each

component featuring a number of interconnected bodies and a number of independent variables, termed local degrees of freedom. For each component, the motion of the bodies of interest is determined as a function of the degrees of freedom at characteristic support points through which a polynomial of limited order (typically of order three) is fitted. These polynomials are then used to interpolate, during simulation, the quantities sought for, avoiding the need to evaluate complicated mathematical expressions from the outset (Wehage and Belczynski 1992, Wehage and Belczynski 1993). It goes without saying that this approach leads to very fast programs. Its only drawback is the exponential grow of the memory and time requirements for preprocessing and subsequent evaluation with the number of degrees of freedom contained in the individual components.

The second alternative is to avoid the phenomenon of intermediate expression swell by taking better account of the kinematic properties of the system at hand. Here, the two issues of importance are the topological and the geometric parameters of the system. The topological parameters comprise information about subsystems into which the system can be dissected and the pattern of the relationships between the dissected subsystems that have to be considered for re-assembly. The knowledge of geometrical parameters, on the other hand, makes it possible to detect and exploit simplifications in the underlying expressions which can lead to performance boosts of a factor of up to 20 when compared to the generic case. By using such expressions at the kinematical level, the dynamic simulation becomes also highly efficient, and the performance of the resulting simulation code become comparable to that of table-lookup methods. The problem with this approach lies in the complicated topological and geometric processing which has to be carried out prior to the evaluation of the dynamical equations. Until very recently, this problem seemed to be so colossal, that virtually no investigator dared to tackle it, not to say to develop a program for dynamic analysis in which such a scheme was implemented.

As a starting point for such a processing, a number of classical and newer results exist in the realm of pure kinematics. For example, it is known that for a single loop featuring six revolute skew joints with unknown rotations, the resulting explicit equation for a single joint variable is a polynomial of degree sixteen (Lee et al. 1990, Raghavan and Roth 1993). However, for many systems with special geometry, as for example loops exhibiting three joints with intersecting or parallel axes, the degree of the polynomial drops to two, and is possible not only to state, but also to solve the closure equation in closed form (Woernle 1988). By applying some notions from the theory of continuous groups, it is possible to design a fully automatic scheme for detecting and generating such closed-form solutions, as was demonstrated in Kecskeméthy and Hiller 1992. The multiple loop case can then be treated by first dissecting the mechanism into a set of independent loops and then assembling the local solutions of the individual loops into a global system of equations that can be solved recursively for many practical cases (Kecskeméthy 1993b). Another technique is to successively remove resolvable segments or so-called *Assur groups* from the

overall system until no bodies remain (Fanghella 1988, Fanghella and Galletti 1993). Howewer, none of these methods have been implemented so far.

## 1.3 Scope of the Work

The purpose of this research is to develop automated procedures that are able to take a general topological and parametric kinematic description of the rigid body part (or of the undeformed deformable bodies) of a vehicle and generate symbolic equations describing the motion of all bodies of interest —at the position, velocity and acceleration level— as functions of the independent joint variables and their first and second time derivatives.

The solution of the problem described above can be regarded to consist of two parts. One part, denoted as the *relative kinematics*, is to determine the relative motion at all joints as functions of the independent joint variables. In the second part, termed the *absolute kinematics*, the motion of the bodies is computed from the known values of the relative motion at all joints. Clearly the difficulty of the project lies in solving the relative kinematics. It is here where the closure of rigid body chains into loops or systems of loops has to be taken into account and the relationships between the motion of the bodies contained in the loops have to be established. This comprises not only the selection of appropriate closure conditions on a loop-by-loop basis, but also the determination of appropriate loops that minimize the effort for establishing the aforementioned relationships. Typically these relationships are highly nested. Thus the main goal of this research is to device an algorithmic approach for gradually dividing a given multibody system into smaller and smaller subsystems, such that, eventually, the subsystems arrived at are either recursively solvable multiple-loop systems, tree-type systems, or clusters of bodies for which the kinematics have to be solved iteratively. In this way, the algorithm is capable of treating each subsystem in the most effective manner, limiting the amount of unnecessary computations to the absolute minimum.

## 1.4 Summary of the Most Important Results

As a result of the research, methods and algorithms for carrying out the following tasks have been developed and made operative under the symbolic programming language *Mathematica* (Wolfram 1988):

1. Automatic dissection of multibody systems into *clusters*, where each cluster is either a single rigid body, or an assembly in which at least two joints have to be removed in order for it to fall apart into two parts. Clusters are thus either single bodies, single loops, or assemblies of loops for which each loop

shares at least one body with at least one other loop. In combination with the modules described below, this module also identifies the recursively and non-recursively solvable parts of a system (Section **3**).

2. For each cluster, automatic determination of a set of *smallest independent loops*, termed also the *minimal cycle basis* (Section **4**).

3. For each loop, detection and generation of closed-form solutions for the cases where this is possible. (Section 5).

4. For each single loop, selection of the most appropriate closure condition and frame of decomposition for generation of velocity and acceleration expressions with minimal computational effort (Section **6**).

5. For explicitly solvable multiple-loop clusters, determination of *kinematical networks* and an appropriate "solution flow", i.e. equation ordering, such that the local kinematics of each loop can be solved recursively (Section **7**).

6. For non-recursively solvable subsystems, determination of appropriate cut-set equations that minimize the computational effort during iteration, and generation of the corresponding position, velocity and acceleration equations (Section **8**).

7. Re-assembly and reordering of all generated expressions into a global system of equations that is suitable for processing using a conventional computer programming language (Section **9**).

8. Application of the developed methods to a non-trivial example which has been selected in conjunction with the US TARDEC in Warren, Michigan (Section **11**).

This report covers only the theoretic foundations of the developed methods as well as several illustrative examples showing the scope and the output of the programs. It does not cover all the details of the implementation of the procedures in *Mathematica*, which the authors shall be glad to deliver upon request.

# 2 Basic Kinematic Relationships

For an automated processing of the kinematics or dynamics of a multibody system, a suitable description of the system structure and parameters is required. This description should be simple, general and it should mirror the way of thinking of an applications engineer. In addition, the system definition statements should be structured in such a way that they can be used directly as an input file for a symbolic processing computer language, such as Mathematica.

In this section, some basic equations and terminology for the kinematics of serial chains are recollected. Moreover, the basic syntax for the definition of kinematic structures and the actual transformations between the bodies shall be discussed. This description language will constitute the basis for the further processing of the kinematical equations.

Kinematics are not only useful for analyzing motion, but also for generating and solving the dynamical equations for mechanical systems of general structure. In fact, kinematics play a key role in the quest for efficient and robust dynamical equations. Although the generation of dynamical equations is not within the scope of this paper, all derived methodologies, schemes and algorithms are of imminent applicability for the dynamics. This shall be discussed in Section **2.1**, where also the basic decoupling of the two main problems of multibody modeling, namely, the problem of determining relative and absolute motions, and the problem of generating the system's dynamical equations, are sifted out.

## 2.1 Kinematical Issues in Multibody Dynamics

The equations of motion of general multibody systems can be stated by making use of d'Alembert's principle, or any other suitable principle of mechanics. For example for a scleronomic, holonomic system consisting of $n_{\mathcal{B}}$ rigid bodies, d'Alembert's principle takes the form (see also Hiller and Kecskemethy 1989):

$$\sum_{i=1}^{n_{\mathcal{B}}} \Big[ (m_i \, \boldsymbol{a}_{S_i} - \boldsymbol{f}_i^e) \cdot \delta \boldsymbol{s}_i + (\boldsymbol{\Theta}_{S_i} \, \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \boldsymbol{\Theta}_{S_i} \, \boldsymbol{\omega}_i - \boldsymbol{\tau}_{S_i}^e) \cdot \delta \boldsymbol{\phi}_i \Big] = 0 \qquad (1)$$

where the symbols introduced above denote, for each body $\mathcal{B}_i$,

| | | |
|---|---|---|
| $m_i$ | - | mass, |
| $\boldsymbol{\Theta}_i$ | - | tensor of mass-inertia, |
| $\boldsymbol{f}_i$ | - | resultant vector of applied forces, |
| $\boldsymbol{\tau}_{S_i}$ | - | resultant vector of applied moments at center of gravity, |
| $\boldsymbol{a}_{S_i} = \ddot{\boldsymbol{s}}_i$ | - | vector of acceleration of center of gravity, |

$\boldsymbol{\omega}_i$ - vector of angular velocity,
$\dot{\boldsymbol{\omega}}_i$ - vector of angular acceleration,
$\delta \boldsymbol{s}_i$ - vector of virtual displacement of center of gravity,
$\delta \boldsymbol{\phi}_i$ - vector of virtual rotation.

The constraint forces do not appear in Eq. (1) as the virtual displacements $\delta \boldsymbol{s}_i$ and $\delta \boldsymbol{\phi}_i$ are assumed to be compatible with all constraints of the system. In general, the virtual displacements $\delta \boldsymbol{s}_i$ and $\delta \boldsymbol{\phi}_i$ are not independent, so one can not use the above equation to perform actual computations. Instead, what needs to be done is to introduce $f$ independent generalized coordinates $\underline{q} = [q_1, \ldots, q_f]^{\mathrm{T}}$, and then to relate these generalized coordinates to the dependent ones in such a way that the constraint equations are fulfilled exactly. This produces the position transmission equations as

$$\left.\begin{array}{rcl} \boldsymbol{s}_i & = & \boldsymbol{s}_i(q_1, \ldots, q_f) \\ \boldsymbol{R}_i & = & \boldsymbol{R}_i(q_1, \ldots, q_f) \end{array}\right\} \; i = 1, 2, \ldots, n_{\mathcal{B}} \; , \tag{2}$$

where $\boldsymbol{R}_i$ denotes the orthogonal matrix measuring the rotation of body $\mathcal{B}_i$ with respect to the inertial frame. These functions are generally not known explicitly, and, in fact, can be determined in general only iteratively from large systems of implicit nonlinear equations. By taking the time-derivative of the positon equations, the velocity transformations

$$\boldsymbol{v}_i = \boldsymbol{J}_{si}\, \underline{\dot{q}} \;\; , \;\; \boldsymbol{\omega}_i = \boldsymbol{J}_{\omega i}\, \underline{\dot{q}} \tag{3}$$

results, which also hold in similar way for the virtual displacements, as

$$\delta \boldsymbol{s}_i = \boldsymbol{J}_{si}\, \delta \underline{q} \;\; , \;\; \delta \boldsymbol{\phi}_i = \boldsymbol{J}_{\omega i}\, \delta \underline{q} \; . \tag{4}$$

By time-differentiation of the velocity transformation equations, one obtains for the acceleration transmission

$$\boldsymbol{a}_{S_i} = \boldsymbol{J}_{si}\, \underline{\ddot{q}} + \dot{\boldsymbol{J}}_{Si}\, \underline{\dot{q}} \;\; , \;\; \dot{\boldsymbol{\omega}}_i = \boldsymbol{J}_{\omega i}\, \underline{\ddot{q}} + \dot{\boldsymbol{J}}_{\omega i}\, \underline{\dot{q}} \; . \tag{5}$$

Insertion of these transformations into d'Alembert's principle yields, due to the independency of virtual displacements $\delta q_1, \ldots, \delta q_f$, the equations of motion of minimal order

$$\boldsymbol{M}\, \underline{\ddot{q}} + \underline{b} = \underline{Q} \; , \tag{6}$$

where the $f \times f$ generalized mass-matrix $\boldsymbol{M}$, the $f \times 1$ matrix of generalized centrifugal and coriolis forces $\underline{b}$ and the $f \times 1$ matrix of generalized applied forces $\underline{Q}$ read, respectively,

$$\boldsymbol{M}\,(\underline{q}) \;=\; \sum_{i=1}^{n_B} \left[ m_i\, \boldsymbol{J}_{s_i}^{\mathrm{T}}\, \boldsymbol{J}_{s_i} + \boldsymbol{J}_{\boldsymbol{\omega}_i}^{\mathrm{T}} \boldsymbol{\Theta}_i \boldsymbol{J}_{\boldsymbol{\omega}_i} \right] \;, \tag{7}$$

$$\underline{b}(\underline{q}\,,\,\dot{\underline{q}}) \;=\; \sum_{i=1}^{n_B} \left[ m_i\, \boldsymbol{J}_{s_i}^{\mathrm{T}}\, \dot{\boldsymbol{J}}_{S_i}\, \dot{\underline{q}} + \boldsymbol{J}_{\boldsymbol{\omega}_i}^{\mathrm{T}} \left( \boldsymbol{\Theta}_i\, \dot{\boldsymbol{J}}_{\omega_i}\, \dot{\underline{q}} \;+ \boldsymbol{\omega}_i \times \boldsymbol{\Theta}_i\, \boldsymbol{\omega}_i \right) \right] \;, \tag{8}$$

$$\underline{Q}(\underline{q}\,,\,\dot{\underline{q}}) \;=\; \sum_{i=1}^{n_B} \left[ \boldsymbol{J}_{s_i}^{\mathrm{T}}\, \boldsymbol{f}_i + \boldsymbol{J}_{\boldsymbol{\omega}_i}^{\mathrm{T}}\, \boldsymbol{\tau}_i \right] \;. \tag{9}$$

In these equations, the only unknown terms are the $3 \times f$ transformation matrices $\boldsymbol{J}_{s_i}$ and $\boldsymbol{J}_{\boldsymbol{\omega}_i}$ and the $3 \times 1$ vectors $\dot{\boldsymbol{J}}_{\omega_i}\, \dot{\underline{q}}$ and $\dot{\boldsymbol{J}}_{S_i}\, \dot{\underline{q}}$. Thus, once these terms are established, the problem of generating the equations of motion of minimal order is reduced to pure algebraic operations. The difficulty for complex multibody systems thus lies in obtaining the transformations of Eq. (3) and Eq. (5). This is the goal of the present project.

## 2.2 Relative and Absolute Kinematics

The transmission equations (3), (4) and (5) involve a large number of equations, of which some are intrinsically implicit, while the others are intrinsically recursive. For each type of equations there exist optimized methods. Thus, it is advantageous to discern between these two types of operations from the outset.

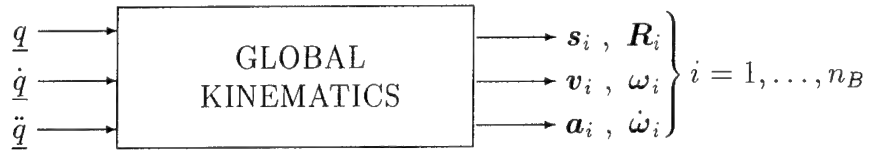



Figure 1: Global Kinematics

Let the overall task of determing the motion of all bodies as functions of the independents be accomplished by a module termed "global kinematics", as depicted in Fig. 1. This task involves on the one hand the determination of the relationships between the relative motion of bodies contained in the same loop and on the other hand the computation of the motion of the bodies as a whole in dependency of the relative motion of the joints. Thus, the task of global kinematics can divided in two sub-tasks, as shown in Fig. 2:

- *relative kinematics*, where all dependent joint variables $\underline{\beta}$ and its derivatives $\dot{\underline{\beta}}, \ddot{\underline{\beta}}$ are expressed as functions of $\underline{q}\,,\, \dot{\underline{q}}\,,\, \ddot{\underline{q}}$.

- *absolute kinematics*, in which, by a forward kinematics procedure, kinematical quantities such as $\boldsymbol{s}_i, \boldsymbol{R}_i, \boldsymbol{v}_i, \boldsymbol{\omega}_i, \boldsymbol{a}_i, \dot{\boldsymbol{\omega}}_i$ are calculated for all bodies as a function of $\underline{\beta}$, $\underline{\dot{\beta}}$, $\underline{\ddot{\beta}}$.

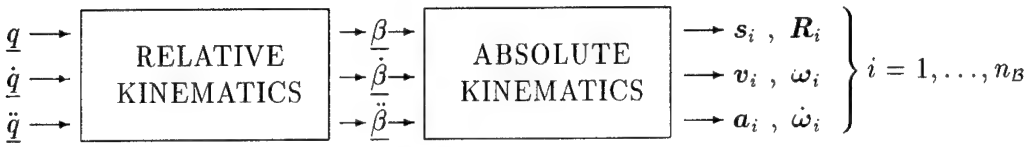



Figure 2: Kinematics of multibody systems with closed loops

In the following, we first concentrate in developing efficient methods for solving the relative kinematics for the case of general multiple-loop systems. This will be the main concern of Sections **3** to **8**. Then, in Section **9**, we use these results to produce the absolute kinematics, which then gives us the global kinematics. The further use of the global kinematics, namely, in the setting of the generation of the dynamical equations, shall not be covered in this project. Such investigations could be the topic of further research projects.

## 2.3 Kinematics of Serial Chains

In this section, the basic formulas for computing the absolute motion of a number of serially connected bodies is regarded. To each body we regard to be attached a frame $\mathcal{K}_i$, which shall be taken as representative for that body. The position and orientation of a reference frame $\mathcal{K}_j$ relative to a reference frame $\mathcal{K}_i$ can be described by a $4 \times 4$ homogeneous transformation matrix ${}^i\mathrm{A}_j$ of the form

$$ {}^i\mathrm{A}_j = \begin{bmatrix} {}^i\boldsymbol{R}_j & {}_i^i\underline{r}_j \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} & r_1 \\ \rho_{21} & \rho_{22} & \rho_{23} & r_2 \\ \rho_{31} & \rho_{32} & \rho_{33} & r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{10} $$

where ${}^i\boldsymbol{R}_j$ is the orthonormal $3 \times 3$ matrix of rotation transforming vector components from $\mathcal{K}_j$ to $\mathcal{K}_i$ and ${}_i^i\underline{r}_j$ is the radius vector from the origin $\mathcal{O}_i$ of $\mathcal{K}_i$ to the origin $\mathcal{O}_j$ of $\mathcal{K}_j$. For a sequence of two homogeneous transformations between three reference frames $\mathcal{K}_i, \mathcal{K}_j$ and $\mathcal{K}_k$ (Fig. 3), the composite transformation can be calculated by multiplying the individual transformations as

$$ {}^i\mathrm{A}_k = {}^i\mathrm{A}_j \, {}^j\mathrm{A}_k \quad . \tag{11} $$

Velocities and accelerations can be calculated by successive combination of known relative motions. For a chain of $n$ *elementary* — i.e. revolute or prismatic — joints

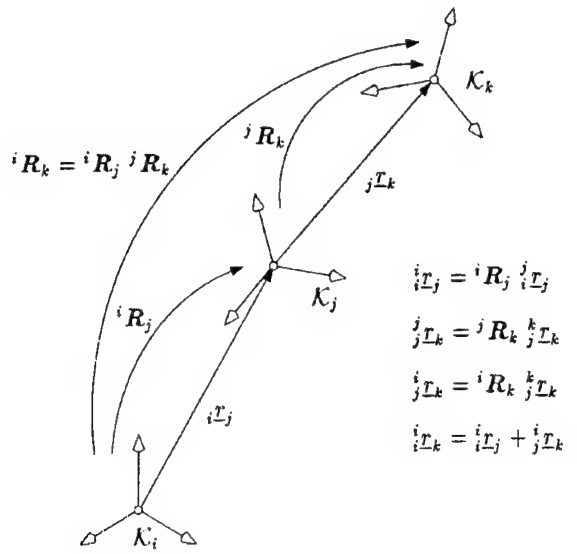



Figure 3: Sequence of transformations

$\mathcal{G}_i$ with joint coordinates $\beta_i$ and joint axes $\boldsymbol{u}_i$, one can calculate the rotational and translational velocity of a reference system $\mathcal{K}_{n+1}$ at the tip of the chain relative to the system $\mathcal{K}_1$ at the bottom as

$$\boldsymbol{\omega}_{n+1} = \sum_1^n \bar{\sigma}_i\, \dot{\beta}_i\, \boldsymbol{u}_i \quad , \tag{12}$$

$$\boldsymbol{v}_{n+1} = \sum_1^n \bar{\sigma}_i\, \dot{\beta}_i\, \boldsymbol{\chi}_i + \sigma_i\, \dot{\beta}_i\, \boldsymbol{u}_i \quad , \quad \boldsymbol{\chi}_i = \boldsymbol{u}_i \times {}_i\boldsymbol{r}_{n+1} \tag{13}$$

where the linear velocity of $\mathcal{K}_n$ is measured with respect to the origin of $\mathcal{K}_n$. Here, $\sigma_i$ denotes a Boolean variable with the values

$$\sigma_i = \begin{cases} 0 & \text{if joint } \mathcal{G}_i \text{ is a revolute joint} \\ 1 & \text{if joint } \mathcal{G}_i \text{ is a prismatic joint} \end{cases} \quad , \tag{14}$$

while $\bar{\sigma}_i = 1 - \sigma_i$ is its complement. Rotational and translational velocity can be combined to a *twist* $\underline{t}_n = [\,\boldsymbol{\omega}_n, \boldsymbol{v}_n]^T$, with which equation (13) becomes

$$\underline{t}_{n+1} = {}^1\boldsymbol{J}_{n+1}\, \underline{\dot{\beta}} \tag{15}$$

with the $6 \times n$ Jacobian (Renaud 1981)

$${}^1\boldsymbol{J}_{n+1} = \begin{bmatrix} \bar{\sigma}_1\, \boldsymbol{u}_1 & \cdots & \bar{\sigma}_n\, \boldsymbol{u}_n \\ \sigma_1\, \boldsymbol{u}_1 + \bar{\sigma}_1\, \boldsymbol{\chi}_1 & \cdots & \sigma_n\, \boldsymbol{u}_n + \bar{\sigma}_n\, \boldsymbol{\chi}_n \end{bmatrix} . \tag{16}$$

The acceleration of $\mathcal{K}_{n+1}$ with respect to $\mathcal{K}_1$ follows from the time-differentiation of equation Eq. (15) as:

$$\dot{\underline{t}}_{n+1} = {}^1\boldsymbol{J}_{n+1}\,\ddot{\underline{\beta}} + {}^1\dot{\boldsymbol{J}}_{n+1}\,\dot{\underline{\beta}} \quad , \tag{17}$$

where the columns of ${}^1\dot{\boldsymbol{J}}_{n+1}$ can be computed as

$$\dot{\boldsymbol{u}}_i = \sum_{j=1}^{i-1} \bar{\sigma}_j\,\dot{\beta}_j\,\boldsymbol{u}_j \times \boldsymbol{u}_i \quad , \tag{18}$$

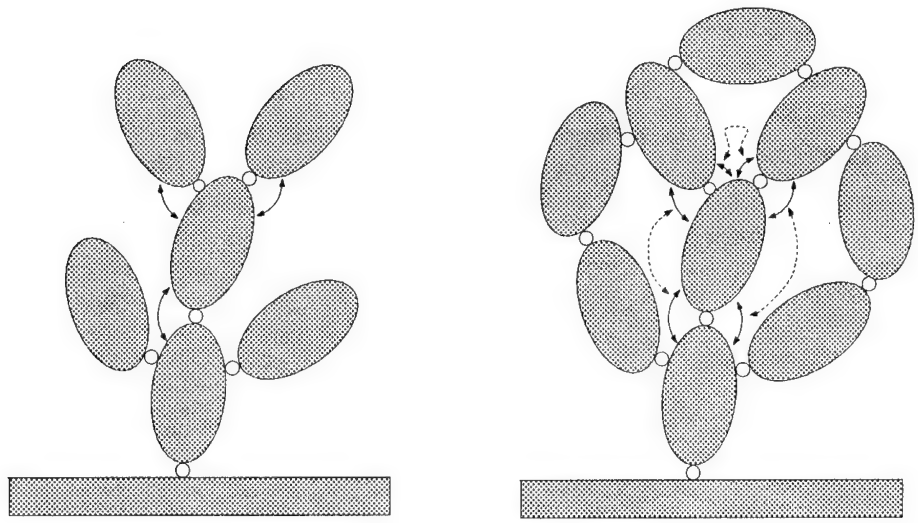
$$\dot{\boldsymbol{\chi}}_i = \sum_{j=1}^{i} \bar{\sigma}_j\,\dot{\beta}_j\,\boldsymbol{u}_j \times \boldsymbol{\chi}_i + \bar{\sigma}_i\,\boldsymbol{u}_i \times \sum_{j=i+1}^{n} \dot{\beta}_j \left(\bar{\sigma}_j\,\boldsymbol{\chi}_j + \sigma_j\,\boldsymbol{u}_j\right) \quad . \tag{19}$$

## 2.4 Topological and Geometric Parameters

As stated in the introduction, the definition of a kinematic structure involves two basic types of information: (1) topological or structural and (2) geometric or parametric information. The topological data comprises information about the number of bodies and joints contained in the system and the interconnection pattern between these. The geometrical data determines the exact location of the axes and points at which relative motions take place, as well as the location of the center of gravity and the attitude of the body-fixed frame. Note that in certain cases geometrical data can come very close to topological information, e. g. when a distance or an angle between two axes become zero.

### 2.4.1 Basic Topological Information

Concerning topological information the existence of loops is a very fundamental property that plays the role of a watershed for the complexity and methodology of the processing algorithms. If the system contains no closed chains, it is said to have a *tree-type* or open structure, while if it contains at least one loop it is termed a *closed-loop system* (Fig. 4). Clearly, for tree-type systems, the relative motion at the joints are independent of one another. Thus, the task of kinematics is limited to computing the absolute motion of the bodies as a function of the relative motions at the joints. On the other hand, when closed loops arise, the relative motions at the joints are not independent anymore and may involve quite complex dependencies. In this case, in addition to the problem of determining the absolute motion of the bodies as functions of the relative joint motions, one has to compute the aforementioned dependencies between the joints motions beforehand. Thus, systems with closed loops involve the two tasks defined above, namely: (1) the determination of joint-motion dependencies, termed the *relative kinematics*, and (2) the determination of absolute motion, termed *absolute kinematics*, while tree-type systems involve only the second task.



a) tree-type system b) closed-loop system

Figure 4: Comparison of tree-type and closed-loop systems

### 2.4.2 Basic Geometric Information

Geometric information regards the knowledge of where the characteristic points, lines and frames through which the bodies interact with other bodies or the environment, and with respect to which other relevant parameters, such as inertia or extension, are measured, are placed within them. Typically, each body will have a body-fixed frame attached to it that acts as a reference for all other quantities measured with respect to that body. Moreover, additional reference frames can be placed on the body when dealing with points or lines, for which a notion of the full spatial pose is required in order to be able to define vectors and matrices. Thus, what regards kinematics, one can think of the multibody system as a collection of reference frames whose relative placement is either constant or time-dependent, and which is defined through constant or variable transformations.

The distinction between topological and geometric information has a subtle point that tends to smear the boundaries between these two notions. For example, when the axes of three rotational joints intersect at one point, is this a geometrical or topological property? Clearly, if the three axes are material, such as in a robot wrist, one would tend to consider this arrangement as a special geometric case, particularly because due to manufacturing errors it might happen that the intersection condition is not fulfilled exactly. On the other hand, if the axes are introduced only as a means of modeling a fixed-point spatial rotation, such as in a ball-and-socket joint of a human hip, then it would be natural to regard the three axes as one topological unit termed a "spherical" joint. Therefore, a kinematic processing engine must always provide a means of extracting relevant topological information

from geometrical one, and vice versa, avoiding a rigid separation of topological and geometrical information that might lead to undetected simplifications.

## 2.5 Description of System Structure

In this work, topological information is stored together with geometrical information by introducing enough reference frames such that any place of interest and any interaction can be modeled by a sequence of elementary transformations. The interconnection between the reference frames is stated in form of a *frame connectivity matrix* $K$, in which each element $K_{ij}$ describes the sequence of elementary or composite transformations that brings the frame $\mathcal{K}_i$ in coincidence with a neighboring frame $\mathcal{K}_j$. Thus, for non-neighboring reference frames the corresponding entries are left empty and the connectivity matrix will typically be a sparse matrix.

The allowed transformations between reference frames are the elementary transformations described in Table 1, as well as their concatenation. An elementary transformation is defined as a template of the form `ETransform[`$k$ , $\sigma$ , $\xi$ `]`, where $k$ represents the coordinate axis about or along which the transformation takes place, $\sigma$ encodes whether the transformations is a rotation ( $\sigma = 0$ ) or a translation ( $\sigma = 1$ ), and $\xi$ is the magnitude of the transformation. A sequence of elementary transformations separated by periods (.) represents the concatenation of the transformations. Note that any transformation can be represented as a sequence of elementary transformations. One particular composite transformation is the Denavit-Hartenberg transformation (Denavit and Hartenberg 1955) that has become very popular in robotics Table 1.

## 2.6 Examples

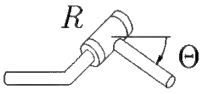
In order to illustrate the concepts described above, we introduce in the following a number of examples, which shall be used also in the subsequent sections.

### 2.6.1 Planar Manipulator

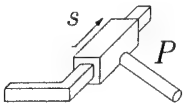
The manipulator shown in Fig. 5 is a planar tree-structured system consisting of two revolute joints and two rigid links. For the definition of system topology, five reference frames, denoted $\mathcal{K}_1, \ldots, \mathcal{K}_5$, have been introduced, where $\mathcal{K}_1$ denotes the basis and $\mathcal{K}_2, \mathcal{K}_3$ and $\mathcal{K}_4, \mathcal{K}_5$ are two pairs of reference frames attached respectively to the arms. Note that the introduction of two reference frames per arm is optional. However, there is no computational penalty for this kind of redundant frame definition.

The pertaining transformations between neighboring reference frames is given by

elementary rotations




elementary translations




$$\texttt{ETransform[1,0,}\Theta\texttt{]} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Theta & -\sin\Theta & 0 \\ 0 & \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\texttt{ETransform[1,1,}s\texttt{]} = \begin{pmatrix} 1 & 0 & 0 & s \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

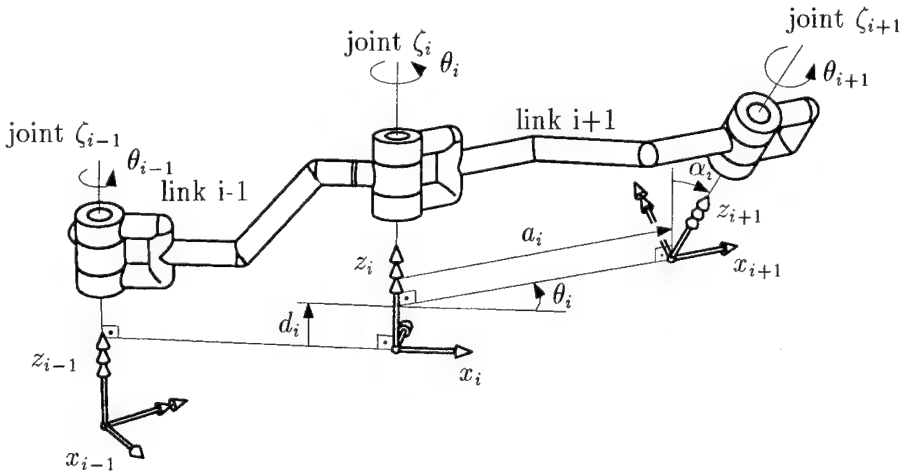
$$\texttt{ETransform[2,0,}\Theta\texttt{]} = \begin{pmatrix} \cos\Theta & 0 & \sin\Theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Theta & 0 & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\texttt{ETransform[2,1,}s\texttt{]} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\texttt{ETransform[3,0,}\Theta\texttt{]} = \begin{pmatrix} \cos\Theta & -\sin\Theta & 0 & 0 \\ \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\texttt{ETransform[3,1,}s\texttt{]} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & s \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Denavit-Hartenberg Parametrisation




$$\texttt{DHTransform[}\Theta,d,\alpha,a\texttt{]} = \texttt{ETransform[3,0,}\theta\texttt{]} \,.\, \texttt{ETransform[3,1,}d\texttt{]} \,.\, \texttt{ETransform[1,0,}\alpha\texttt{]} \,.\, \texttt{ETransform[1,1,}a\texttt{]}$$

$$= \begin{pmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & a\cos\theta \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & a\sin\theta \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Table 1: Macros for description of basic transformations

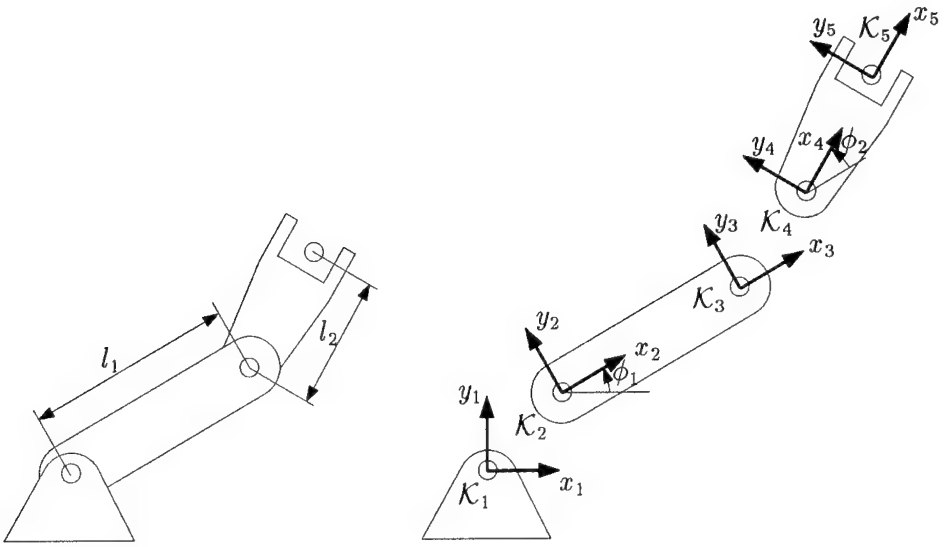



Figure 5: Planar manipulator with two axes

```
K=Table[Null,{5},{5}];
K[[1,2]] = ETransform[3,0,phi[1]];
K[[2,3]] = ETransform[1,1,l[1]];
K[[3,4]] = ETransform[3,0,phi[2]];
K[[4,5]] = ETransform[1,1,l[2]];
```

Note that for `K` we are using the *Mathematica* convention for matrix elements involving double square brackets, while for `phi` and `l` we use only simple brackets. This is because the latter quantities are not regarded as matrices, but represent just particular names of variables. Note also that it is not necessary to prescribe also the inverse of a given transformation, as these are generated automatically by the code

### 2.6.2 Wheel Suspension of a Trailer

As a second example, consider the mechanism depicted in Fig. 6, which corresponds to a wheel suspension of a trailer. Here, eight reference frames are introduced: three at the rod, three at the lower arm and one at each member of the shock absorber. The pertaining transformations between neighboring reference frames are

```
K=Table[Null,{8},{8}];
K[[1,2]] = ETransform[2,1,v1y];
K[[1,3]] = ETransform[2,1,v2y].ETransform[3,1,-v2z];
K[[2,4]] = ETransform[1,0,alpha];
K[[4,5]] = ETransform[2,1,-v3y].ETransform[3,1,v3z];
```

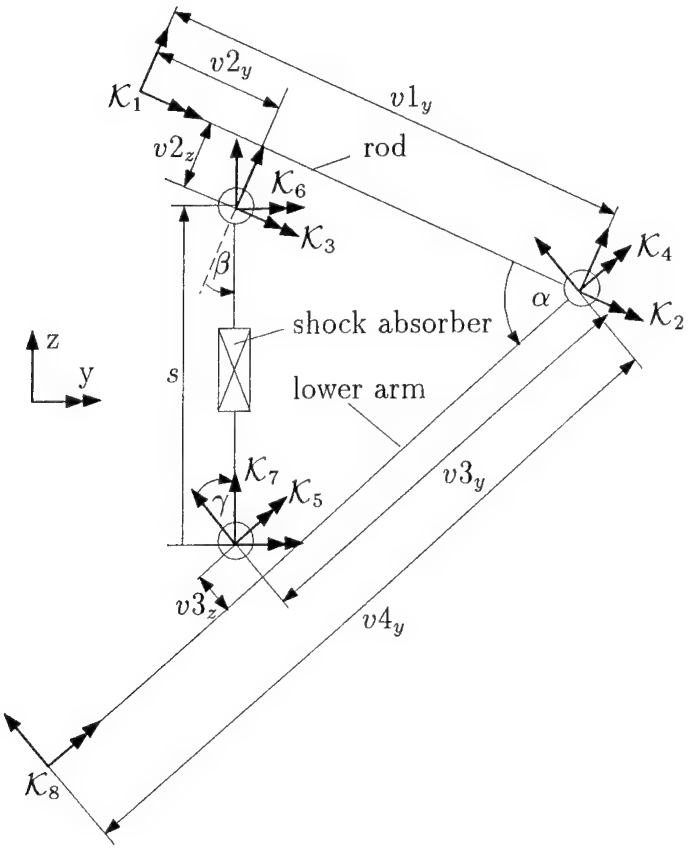



Figure 6: Wheel suspension of a trailer

```
K[[3,6]] = ETransform[1,0,beta];
K[[7,6]] = ETransform[3,1,s];
K[[5,7]] = ETransform[1,0,gamma];
K[[4,8]] = ETransform[2,1,-v4y];
```

Note that the here some composite transformations have been use for selected elements of the matrix `K`.

### 2.6.3 A Heavy-Load Manipulator

A more involved example is the heavy load manipulator illustrated in Fig. 7, which is currently under investigation at the Department of Mechatronics of the Gerhard Mercator University of Duisburg (Schneider and Hiller 1995). The system consists of several booms that are connected together through joint assemblies featuring several closed loops (Fig. 8). Each boom-joint assembly represents a *joint unit* whose kinematics can be analyzed separately. The complete system consists of one "shoulder" joint and two additional "elbow" joints. More involved systems, such as those used for aircraft washing applications, comprise up to five joint units.

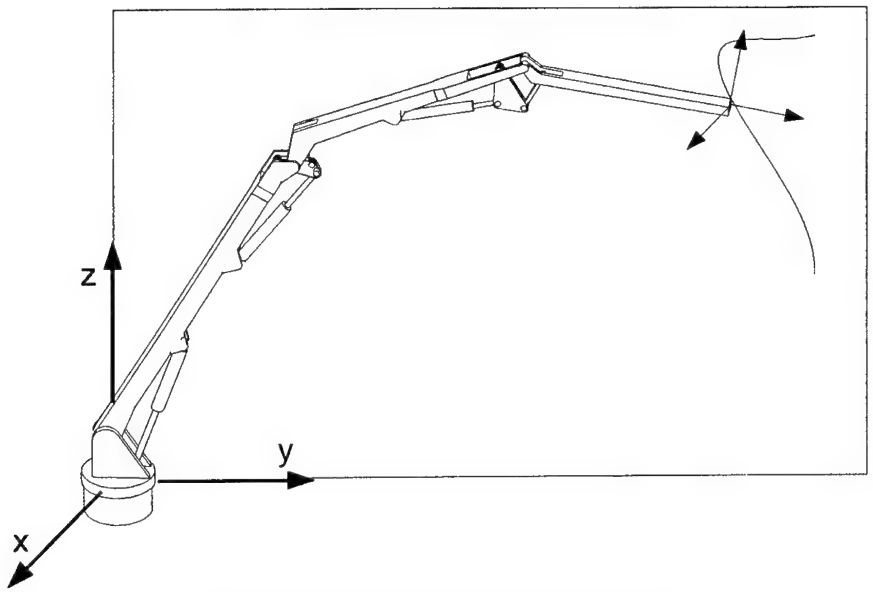



Figure 7: A heavy-load manipulator

For the definition of the kinematic structure, 33 reference frames are introduced, which are numbered from 1 to 33. Between these frames, constant and variable transformations are introduced. Constant transformation parameters (mostly lengths) are denoted by `v[i,j]`, where `i` denotes the index of the current joint unit and `j` is a running index within that joint unit. Variable joint parameters are denoted by `beta[i,j]` in the rotational case and `s[i]` in the translational case. The statements defining the kinematical structure are reproduced below. For better clarity, an iconic model for each joint unit is supplied in Fig. 9.

```
K=Table[Null,{33},{33}];
K[[1,2]]   = ETransform[3,1,v[1,1]];
K[[1,3]]   = ETransform[3,1,v[1,2]].ETransform[2,1,v[1,3]];
K[[3,4]]   = ETransform[1,0,beta[1,2]];
K[[4,5]]   = ETransform[3,1,s[1]];
K[[5,6]]   = ETransform[1,0,beta[1,3]];
K[[2,7]]   = ETransform[1,0,beta[1,1]];
K[[7,6]]   = ETransform[3,1,v[1,4]].ETransform[2,1,v[1,5]];
K[[7,8]]   = ETransform[3,1,v[1,6]];
K[[8,9]]   = ETransform[2,1,v[2,1]];
K[[8,10]]  = ETransform[3,1,v[2,2]];
K[[8,11]]  = ETransform[3,1,v[2,3]].ETransform[2,1,v[2,4]];
K[[9,12]]  = ETransform[1,0,beta[2,2]];
K[[12,13]] = ETransform[3,1,s[2]];
K[[13,14]] = ETransform[1,0,beta[2,5]];
K[[11,15]] = ETransform[1,0,beta[2,3]];
K[[15,14]] = ETransform[3,1,v[2,5]];
K[[15,16]] = ETransform[3,1,v[2,6]].ETransform[2,1,v[2,7]];
```

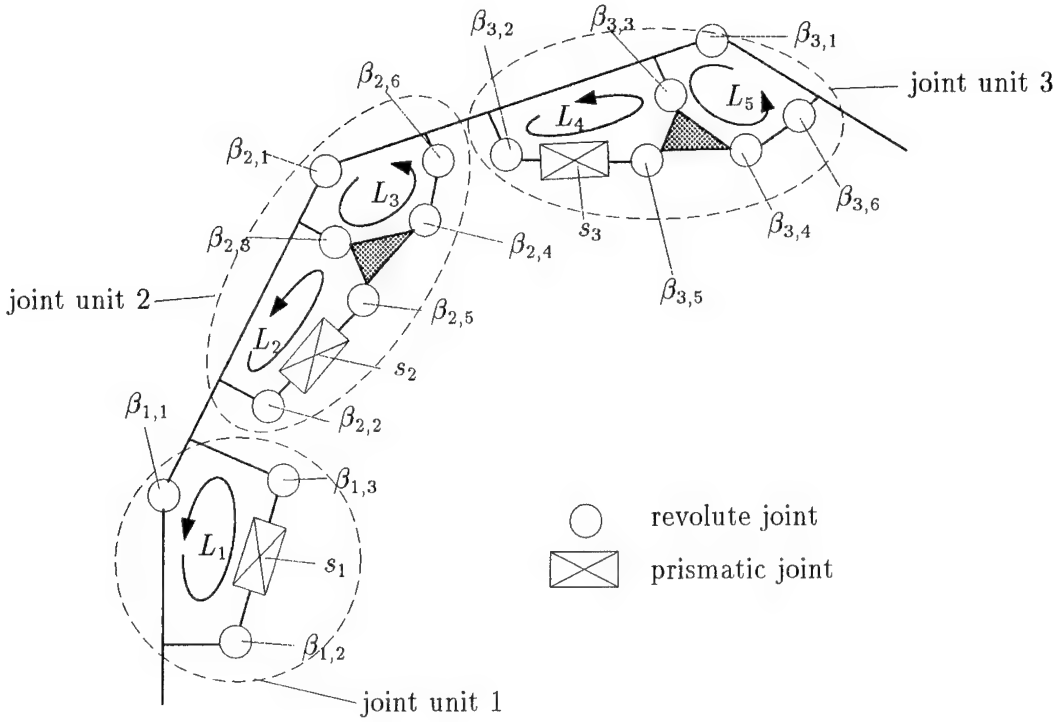



Figure 8: Kinematic structure of the heavy-load manipulator

```
K[[16,17]] = ETransform[1,0,beta[2,4]];
K[[17,18]] = ETransform[3,1,v[2,8]];
K[[18,19]] = ETransform[1,0,beta[2,6]];
K[[10,20]] = ETransform[1,0,beta[2,1]];
K[[19,20]] = ETransform[2,1,v[2,9]].ETransform[3,1,v[2,10]];
K[[20,21]] = ETransform[3,1,v[2,11]];
K[[21,22]] = ETransform[2,1,v[3,1]];
K[[21,23]] = ETransform[3,1,v[3,2]];
K[[21,24]] = ETransform[3,1,v[3,3]].ETransform[2,1,v[3,4]];
K[[22,25]] = ETransform[1,0,beta[3,2]];
K[[25,26]] = ETransform[3,1,s[3]];
K[[26,27]] = ETransform[1,0,beta[3,5]];
K[[24,28]] = ETransform[1,0,beta[3,3]];
K[[28,27]] = ETransform[3,1,v[3,5]];
K[[28,29]] = ETransform[3,1,v[3,6]].ETransform[2,1,v[3,7]];
K[[29,30]] = ETransform[1,0,beta[3,4]];
K[[30,31]] = ETransform[3,1,v[3,8]];
K[[31,32]] = ETransform[1,0,beta[3,6]];
K[[23,33]] = ETransform[1,0,beta[3,1]];
K[[32,33]] = ETransform[3,1,v[3,9]].ETransform[2,1,v[3,10]];
```

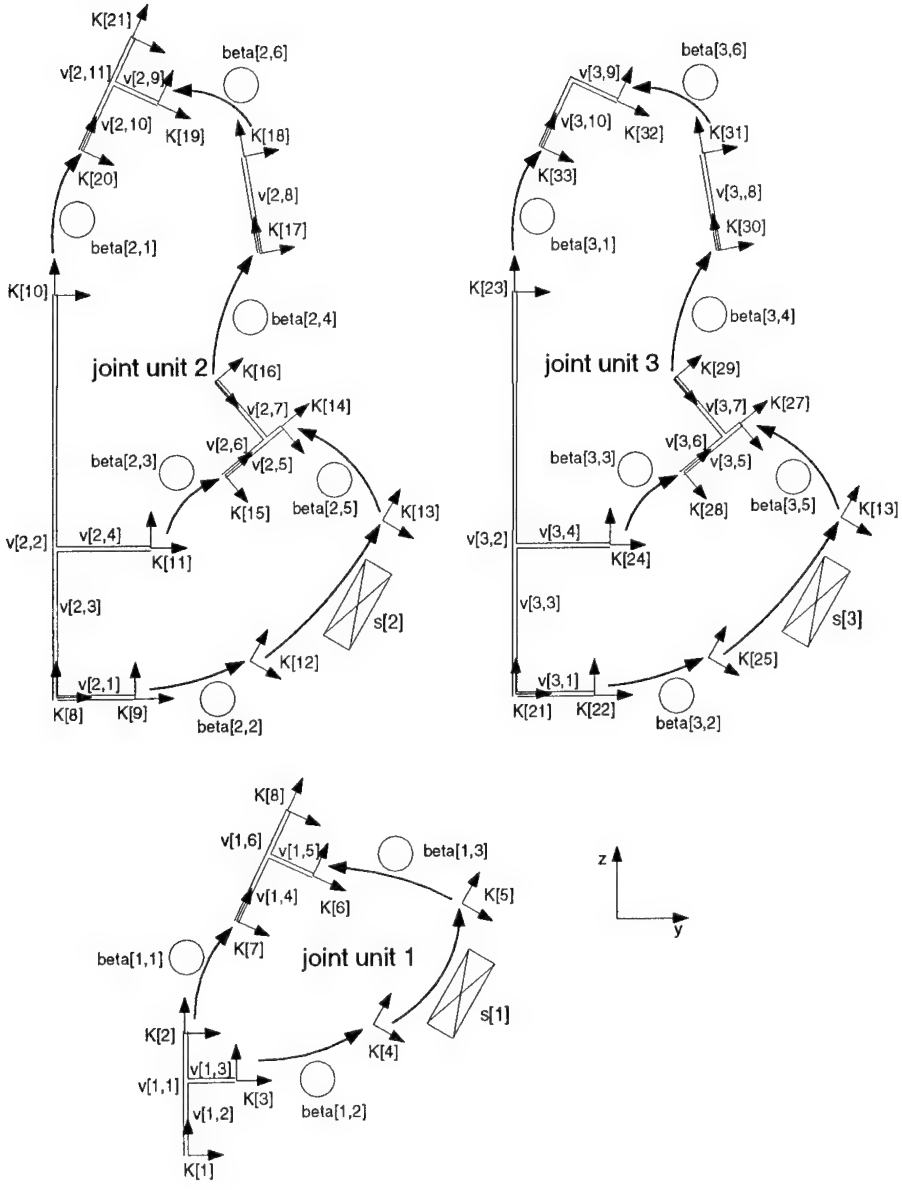



Figure 9: Iconic model of the heavy-load manipulator

# 3 Determination of Clusters

As was mentioned in the previous section, the structure of a mechanism can have significant impact on the complexity of the tasks involved in its processing. In particular, it is crucial to recognize closed-loop and tree-type subsystems, as this implies whether the relative kinematics have to be established or not. The distinction between closed loop subsystems and those that form open chains can be made by regarding the number of joints that have to be removed in order for the remaining subsystem to fall apart in two parts. If removal of a single joint induces a decomposition of the system in two parts, then this joint is part of a tree type substructure. Otherwise the joint is part of a closed-loop subsystem, which shall be termed a "cluster" in the sequel. Tree-type subsystems and clusters can be automatically detected by application of some concepts of graph theory, as described next.

By a (linear) graph $G = \langle K, E \rangle$ one understands a set of nodes $K = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ and a set of directed edges $E = \{e_{i_1,j_1}, \ldots, e_{i_m,j_m}\}$, where each edge $e_{i_k,j_k}$ connects two nodes $\mathcal{K}_{i_k}$ and $\mathcal{K}_{j_k}$, starting at $\mathcal{K}_{i_k}$ and ending at $\mathcal{K}_{j_k}$. The frame connectivity matrix can be easily mapped into a directed graph. Hereby, the reference frames are regarded as nodes and the transformations between them as directed edges. That is, if a transformation from a reference frame $\mathcal{K}_i$ to a reference frame $\mathcal{K}_j$ is given, then in the associated graph one has two nodes $\mathcal{K}_i$ and $\mathcal{K}_j$ corresponding to the two reference frames and an edge $e_{ij}$ directed from $\mathcal{K}_i$ to $\mathcal{K}_j$ representing the transformation.

For example, the graph associated with the heavy-load manipulator has the structure depicted in Fig. 10. Note that in this graph the edges are oriented in accordance with the direction of the user-defined transformations.

A well-known method for detecting the clusters of a graph is to search for the so-called *bridges*, which are edges $e_{ij}$ for which, after removing one of them, the corresponding end nodes $\mathcal{K}_i$ and $\mathcal{K}_j$ are not connected anymore (Carré 1979). An algorithm that is particularly suitable for performing this consists in calculating the so-called "closure" of the adjacency matrix of the graph. In this setting, the aforementioned clusters are termed "leaves".

The algorithm starts with the so-called adjacency matrix $A$ of the graph $G$, the coefficients of which are defined as

$$A_{ij} = \begin{cases} \{e_{ij}\}\,, i < j & \text{if } e_{ij} \in E \text{ or } e_{ji} \in E \\ 0 & \text{if } e_{ij}, e_{ji} \notin E \end{cases} \quad . \tag{20}$$

Hereby, the special element " 0 " is defined to have the following properties when used in the set-theoretic operations of union (" $\cup$ ") and intersection (" $\cap$ ")

$$\begin{aligned} \{e_{ij}, e_{kl}, \ldots\} \cap 0 = 0 \cap \{e_{ij}, e_{kl}, \ldots\} = \{e_{ij}, e_{kl}, \ldots\} \quad , \\ \{e_{ij}, e_{kl}, \ldots\} \cup 0 = 0 \cup \{e_{ij}, e_{kl}, \ldots\} = 0 \quad . \end{aligned} \tag{21}$$

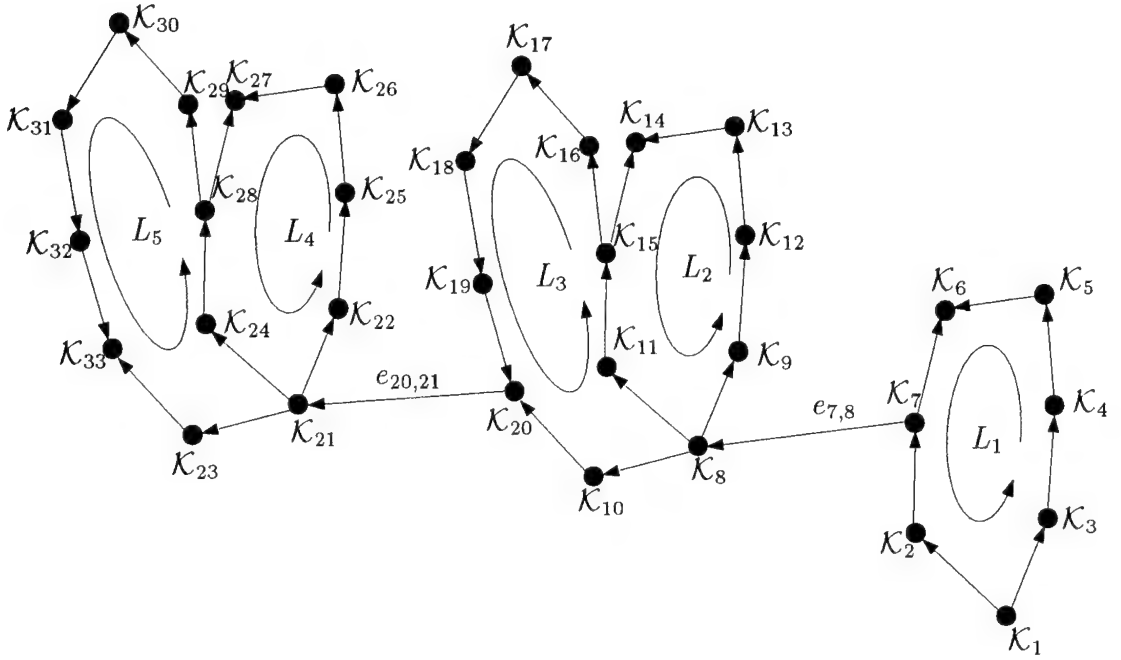



Figure 10: Interconnecting graph for the heavy-load manipulator

Note that the zero element, "0", is different from the empty set, $\emptyset$. In fact, it plays the role of the *complement* of $\emptyset$, i. e., it behaves as the set of all nodes. Starting from the adjacency matrix, a recursive algorithm is applied that generates as coefficients of the resulting matrices *sets* of edges $\{e_{ij}, e_{kl}, \ldots\}$ by taking particular unions and intersections of the coefficients of previous matrices and the zero element defined above. The recursive algorithm corresponds to a simplified Jordan elimination and reads

$$B^{(0)} = A \ , \tag{22}$$

$$B_{ij}^{(k)} = \begin{cases} B_{ij}^{(k-1)} & \text{if } i = k \text{ or } j = k \\ B_{ij}^{(k-1)} \cap \left( B_{ik}^{(k-1)} \cup B_{kj}^{(k-1)} \right) & \text{if } i, j \neq k \end{cases} \ , \ k = 1, 2, \ldots, n \ , \tag{23}$$

where $n$ denotes the number of nodes of $\mathcal{K}$. The final matrix, $B^{(n)}$, obtained in the step $k = n$ represents the so-called *closure* matrix $A^* = B^{(n)}$, of the initial matrix $A$. The closure matrix $A^*$ has the following properties.

(1) If an element of $A_{ij}^*$ of the resulting matrix $A^*$ is equal to the zero element introduced above, or equal to the empty set, the corresponding nodes $\mathcal{K}_i$ and $\mathcal{K}_j$ are part of the same leaf.

(2) If an element $A_{ij}^*$ is equal to a non-empty set of edges, these edges represent the bridges that have to be traversed in order to travel from node $\mathcal{K}_i$ to node $\mathcal{K}_j$.

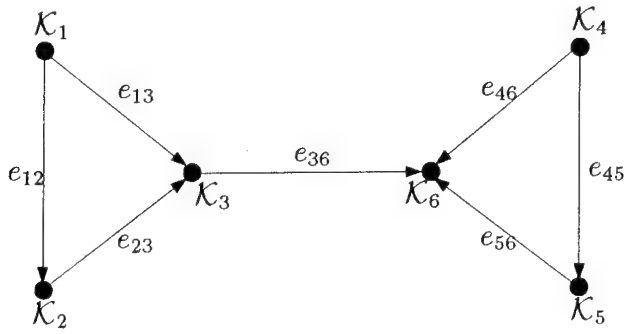



Figure 11: Example of a graph containing two leaves and one bridge

Thus, the coefficients of the closure matrix $A^*$ yield directly the information needed to detect the clusters in a multibody system.

As a simple example, regard the graph displayed in Fig. 11. Its representation as a set of nodes and edges is

$$G = \langle \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_4, \mathcal{K}_5, \mathcal{K}_6\} \ , \ \{e_{12}, e_{13}, e_{23}, e_{36}, e_{45}, e_{46}, e_{56}\} \rangle \ .$$

By immediate inspection, one can recognize that the graph contains two leaves, one leaf consisting of nodes $\{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3\}$ and the other consisting of nodes $\{\mathcal{K}_4, \mathcal{K}_5, \mathcal{K}_6\}$, and that the bridge connecting these two leaves is $e_{36}$.

The algorithm described above recognizes these leaves by processing the corresponding adjacency matrix

$$A = \begin{bmatrix} 0 & \{e_{12}\} & \{e_{13}\} & 0 & 0 & 0 \\ \{e_{12}\} & 0 & \{e_{23}\} & 0 & 0 & 0 \\ \{e_{13}\} & \{e_{23}\} & 0 & 0 & 0 & \{e_{36}\} \\ 0 & 0 & 0 & 0 & \{e_{45}\} & \{e_{46}\} \\ 0 & 0 & 0 & \{e_{45}\} & 0 & \{e_{56}\} \\ 0 & 0 & \{e_{36}\} & \{e_{46}\} & \{e_{56}\} & 0 \end{bmatrix} \ .$$

Form this matrix, one obtains by application of the algorithm described above the sequence of matrices

$$B^{(1)} = \begin{bmatrix} 0 & \{e_{12}\} & \{e_{13}\} & 0 & 0 & 0 \\ \{e_{12}\} & 0 \cap (\{e_{12}\} \cup \{e_{12}\}) = \{e_{12}\} & \{\} & 0 & 0 & 0 \\ \{e_{13}\} & \{e_{23}\} \cap (\{e_{13}\} \cup \{e_{12}\}) = \{\} & \{e_{13}\} & 0 & 0 & \{e_{36}\} \\ 0 & 0 \cap (0 \cup \{e_{12}\}) = 0 & 0 & 0 & \{e_{45}\} & \{e_{46}\} \\ 0 & 0 & 0 & \{e_{45}\} & 0 & \{e_{56}\} \\ 0 & 0 & \{e_{36}\} & \{e_{46}\} & \{e_{56}\} & 0 \end{bmatrix} \ ,$$

$$B^{(2)} = \begin{bmatrix} \{e_{12}\} & \{e_{12}\} & \{\} & 0 & 0 & 0 \\ \{e_{12}\} & \{e_{12}\} & \{\} & 0 & 0 & 0 \\ \{\} & \{\} & \{\} & 0 & 0 & \{e_{36}\} \\ 0 & 0 & 0 & 0 & \{e_{45}\} & \{e_{46}\} \\ 0 & 0 & 0 & \{e_{45}\} & 0 & \{e_{56}\} \\ 0 & 0 & \{e_{36}\} & \{e_{46}\} & \{e_{56}\} & 0 \end{bmatrix},$$

$$B^{(3)} = \begin{bmatrix} \{\} & \{\} & \{\} & 0 & 0 & \{e_{36}\} \\ \{\} & \{\} & \{\} & 0 & 0 & \{e_{36}\} \\ \{\} & \{\} & \{\} & 0 & 0 & \{e_{36}\} \\ 0 & 0 & 0 & 0 & \{e_{45}\} & \{e_{46}\} \\ 0 & 0 & 0 & \{e_{45}\} & 0 & \{e_{56}\} \\ \{e_{36}\} & \{e_{36}\} & \{e_{36}\} & \{e_{46}\} & \{e_{56}\} & \{e_{36}\} \end{bmatrix},$$

$$B^{(4)} = \begin{bmatrix} \{\} & \{\} & \{\} & 0 & 0 & \{e_{36}\} \\ \{\} & \{\} & \{\} & 0 & 0 & \{e_{36}\} \\ \{\} & \{\} & \{\} & 0 & 0 & \{e_{36}\} \\ 0 & 0 & 0 & 0 & \{e_{45}\} & \{e_{46}\} \\ 0 & 0 & 0 & \{e_{45}\} & \{e_{45}\} & \{\} \\ \{e_{36}\} & \{e_{36}\} & \{e_{36}\} & \{e_{46}\} & \{\} & \{\} \end{bmatrix},$$

$$B^{(5)} = \begin{bmatrix} \{\} & \{\} & \{\} & 0 & 0 & \{e_{36}\} \\ \{\} & \{\} & \{\} & 0 & 0 & \{e_{36}\} \\ \{\} & \{\} & \{\} & 0 & 0 & \{e_{36}\} \\ 0 & 0 & 0 & \{e_{45}\} & \{e_{45}\} & \{\} \\ 0 & 0 & 0 & \{e_{45}\} & \{e_{45}\} & \{\} \\ \{e_{36}\} & \{e_{36}\} & \{e_{36}\} & \{\} & \{\} & \{\} \end{bmatrix}.$$

Finally, after $n = 6$ steps the closure matrix

$$A^* = B^{(6)} = \begin{bmatrix} \{\} & \{\} & \{\} & \{e_{36}\} & \{e_{36}\} & \{e_{36}\} \\ \{\} & \{\} & \{\} & \{e_{36}\} & \{e_{36}\} & \{e_{36}\} \\ \{\} & \{\} & \{\} & \{e_{36}\} & \{e_{36}\} & \{e_{36}\} \\ \{e_{36}\} & \{e_{36}\} & \{e_{36}\} & \{\} & \{\} & \{\} \\ \{e_{36}\} & \{e_{36}\} & \{e_{36}\} & \{\} & \{\} & \{\} \\ \{e_{36}\} & \{e_{36}\} & \{e_{36}\} & \{\} & \{\} & \{\} \end{bmatrix}$$

is calculated. Clearly, according to the criteria stated above, nodes $\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3$ and $\mathcal{K}_4, \mathcal{K}_5, \mathcal{K}_6$ belong, respectively, to different leaves, which are separated by the "bridge" $e_{36}$.

Note that for the wheel suspension of (Section **2.6.2**) no decomposition into simply connected subsystems is possible, because the complete system forms one single cluster. However, for the heavy manipulator described in Section **2.6.3**, three leaves separated by the two bridges $e_{7,8}$ and $e_{20,21}$ respectively, can be detected.

# 4 Detection of Independent Kinematical Loops

After applying the dissection algorithm described above, the system is divided into a set of clusters whose relative kinematics are independent of one another. For each cluster, there are two possibilities: (1) it consists of a single loop, or (2) it consists of several interconnected loops.

In the case that the cluster comprises only a single loop, the methods derived in Section ?? can be applied directly.

In the case that the cluster consists of several interconnected loops, one needs to take into account the local kinematics at each loop and the relationships between them. Obviously, the single loop plays a key role in the generation and solution of the relative kinematics. As will be seen later, it is possible to build the complete kinematics of a multi-loop system by assembling the local kinematics of the individual loops in what is termed a "kinematical network". Thus, the recognition and processing of an appropriate set of loops is crucial for the design of a corresponding algorithm. Below we describe a method for automatically recognizing a set of suitable independent loops, which is based on graph-theoretic concepts.

## 4.1 Fundamentals of Cycle Bases

According to a well-known theorem, for a multiple-loop system whose associated graph contains $n_V$ nodes and $n_E$ edges there are $n_L = n_E - n_K + 1$ independent loops (Gondran and Minoux 1984). While the number of independent loops is unique, this is not the case for the set of loops itself, as there are infinitely many ways of choosing a suitable set of $n_L$ loops where no loop results as a combination of the other loops. It is obvious that, depending on the choice of loops, the kinematics may be more or less simple to solve. Thus, it is important to determine a set of loops which simplifies the resolution of the kinematics.

A heuristic criterion for defining a suitable set of loops is that the loops should be as "small" as possible, where *small* refers to the number of joint coordinates contained in the joints of loop. As will be shown in Section **7**, the sum of degrees of freedom over all loops minus the number of coupling conditions between the loops is an invariant of the system which represents its number of degrees of freedom. Thus, if a small loop is replaced by a larger loop, the number of coupling conditions between this loop and the others must increase, and the overall resolution of equations becomes more complicated. Therefore as rule of thumb, smaller loops imply less coupling and thus simpler equations.

The determination of the smallest set of loops corresponds to the problem of finding the *minimal cycle basis* of a graph. This problem is well-known in graph theory, and there have been several methods derived for its solution. Concerning terminology,

there is a subtle difference between the notion of a loop in multibody dynamics and that of a cycle in graph theory. By a multibody loop we understand a serial chain of bodies in which each body is connected to exactly two other bodies. In contrast to this, a cycle is a set of edges and nodes such that each node is connected to an even number of edges. Thus, a cycle can form an "eight", embrace more than one multibody loop, etc. We will exclude these "paradoxical" cases for simplicity and regard only cycles that correspond to single multibody loops. Such cycles are also termed "elementary" cycles in the pertaining literature.

For the operation with cycles, one can define a corresponding algebra as follows. The sum of two cycles $C$ and $D$ is defined as their symmetric difference $C + D = (C \cup D) - (C \cap D)$ . The resulting cycle comprises the union of the edges and nodes of both cycles, minus the common edges and the nodes that are incident to two common edges. The set of cycles in a graph is closed under the operation of addition. A set of cycles is termed *independent* if no cycle in this set results from the addition of other cycles in this set. A set of cycles is a *basis* if any cycle which is not an element of this set can be generated by the sum of some of the cycles in this set.

## 4.2 Methods for Determination of Cycle Bases

A special kind of cycle basis is the fundamental cycle set. In a fundamental cycle set, it is possible to find one edge for each cycle such that removal of all of these edges leads to a connected tree, and re-insertion of each of these edges closes exactly one cycle, which is the one to which the edge is associated (Horton 1987). The remaining graph is termed the *spanning tree* of the graph, and the set of removed edges is denoted the *cotree* (Andrews and Kesavan 1975). Clearly, a cycle basis in which there exists one cycle whose every edge is also part of some other cycle can not be a fundamental cycle basis, because removal of any of the edges of the internal cycle will also open the surrounding cycles. Thus, fundamental cycle sets are special cases of cycle bases, and the set of smallest cycles also termed the *minimal cycle basis* may yield smaller cycles then a corresponding fundamental cycle set.

While polynomial-time algorithms for the determination of minimal cycle bases exist (Hubicka and Syslo 1975, Horton 1987), the problem of determining a fundamental set of minimal cycles is solvable only by trying all possibilities of cycle-forming and comparing them. In the present modeling, it is not required that the multibody loops form a fundamental cycle set. Thus it is possible to use the minimal cycle-basis algorithms for determining a suitable set of loops. Below we state an algorithm that solves this problem in an efficient manner.

## 4.3 Proposed Algorithm for Finding a Minimal Cycle Basis

The present algorithm for determining a minimal cycle basis is a modification of the algorithm described in Horton 1987. Its flow diagram is shown in Fig. 13. We start with the interconnection graph $G$, in which each existing edge is complemented by a corresponding edge in opposite direction. The hereby resulting set of edges is termed $\hat{E}$. The target is to extract from this representation a set of minimal cycles $c_i$ which will be collected in a set $C = \{c_1, \ldots, c_{n_L}\}$. During the process of finding this set of cycles, some bookkeeping needs to be done concerning the shortest paths detected so far. This information is stored in a "shortest-path matrix", $P$, in which each element $p_{ij}$ describes the shortest known path between two nodes $\mathcal{K}_i$ and $\mathcal{K}_j$, stored as a set of edges.

A node and an edge will be marked as "visited" when they are touched by the algorithm; a node will be marked as "finished" when all edges with which it is incident are marked as "visited". At the beginning of the algorithm, all nodes and edges are marked as "unvisited" and "unfinished". The node that is going to constitute the root of the graph is marked as "visited". Furthermore, the shortest-path matrix $P$ is initialized as follows:

$$P_{ij} = \begin{cases} \{e_{ij}\} & \text{if } e_{ij} \in \hat{E} \\ 0 & \text{if else} \end{cases} . \tag{24}$$

Finally, the container for the minimal cycle set is initialized as the empty set $C = \{\}$.

The algorithm consists in the repeated application of the following two steps, each application of a step or of a substep hereof being termed a "stage":

1. Take a node that is marked as "visited" and "unfinished" as the actual node $\mathcal{K}_k$. Then, for each edge $e_{kj}$ to a node $\mathcal{K}_j$ which is not visited, mark edges $e_{kj}$ and $e_{jk}$ as well as node $\mathcal{K}_j$ as visited, and apply the four steps described below. Afterwards, mark the actual node as "finished".

2. The following four steps will be applied for each $e_{kj}$ and $\mathcal{K}_j$ described above:

   2.1. Create paths $p_{mj}$ from each visited node $\mathcal{K}_m, m \neq j, k$ to $\mathcal{K}_j$, and vice versa, by appending the edge $e_{kj}$ to the path from $\mathcal{K}_m$ to $\mathcal{K}_j$ as

   $$p_{mj} = p_{mk} \cup \{e_{kj}\} \tag{25}$$

   and accordingly for the path $p_{jm}$.

   2.2. Check whether an edge $e_{mj}$ already existed. If this is not the case go back to *step 1.* Otherwise, the edge $e_{kj}$ closes a cycle with edges

   $$c_n = p_{mk} \cup \{e_{kj}, e_{jm}\} \tag{26}$$

This cycle can now be appended to the current set of independent cycles $C = \{c_1, \ldots, c_{n-1}\}$, as it is independent of all other cycles $c_1, \ldots, c_{n-1}$ determined so far. The independency follows from the fact that it is the only one containing the edge $e_{jm}$. Moreover it is minimal in the visited part of the graph because it is calculated with the shortest path $p_{mk}$. After generating this cycle, mark the edges $e_{jm}$ and $e_{mj}$ as visited.

2.3. Check whether the shortest paths in the visited part of the graph can be shortened by including the edges $e_{jm}$ or $e_{mj}$ defined above. This is the most elaborate part of the algorithm, and consists of taking each pair of visited nodes $\mathcal{K}_u$ and $\mathcal{K}_v$ and comparing, as shown in Fig. 12, the lengths of $\boxed{1}$ the existing path between $\mathcal{K}_u$ and $\mathcal{K}_v$ with $\boxed{2}$ that from $\mathcal{K}_u$ over $\mathcal{K}_m$ and $\mathcal{K}_j$ to $\mathcal{K}_v$, and $\boxed{3}$ that of the path from $\mathcal{K}_u$ over $\mathcal{K}_j$ and $\mathcal{K}_m$ to $\mathcal{K}_v$. From these, the shortest path is retained.

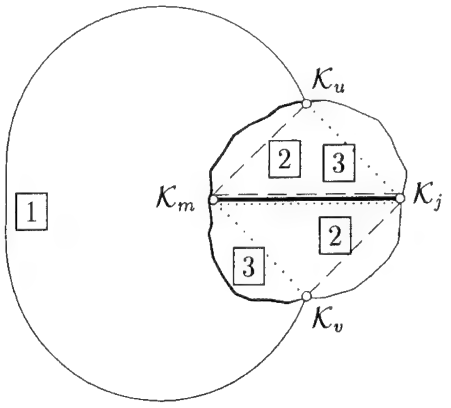



Figure 12: Extraction of shortest paths

2.4. Actualize the set of minimal cycles according to the newly established shortest paths. Hereby, the actual cycle $c_n$ is compared which the result of adding it to one of the previously established cycles $c_j$, $j < n$. If the cycle resulting from $c_n + c_j$ is smaller than the cycle $c_n$, than the latter is replaced by the former.

The described algorithm is displayed as a flow diagram in Fig. 13. It must be emphasized that the present method is only approximative, i. e., it does not guarantee to detect *always* the minimal cycle basis. However, we have compared the algorithm with existing ones and found that it represents a good compromise between efficiency and accuracy. For example, the computational effort of the present approach is $n_K^3$, while for the exact approach described in Horton 1987 it is $n_K^7$. Moreover, we have found that our algorithm renders in general better results than the approximative method described in the same paper, but with almost no additional computational effort. Moreover, the algorithm is easily adapted to the case

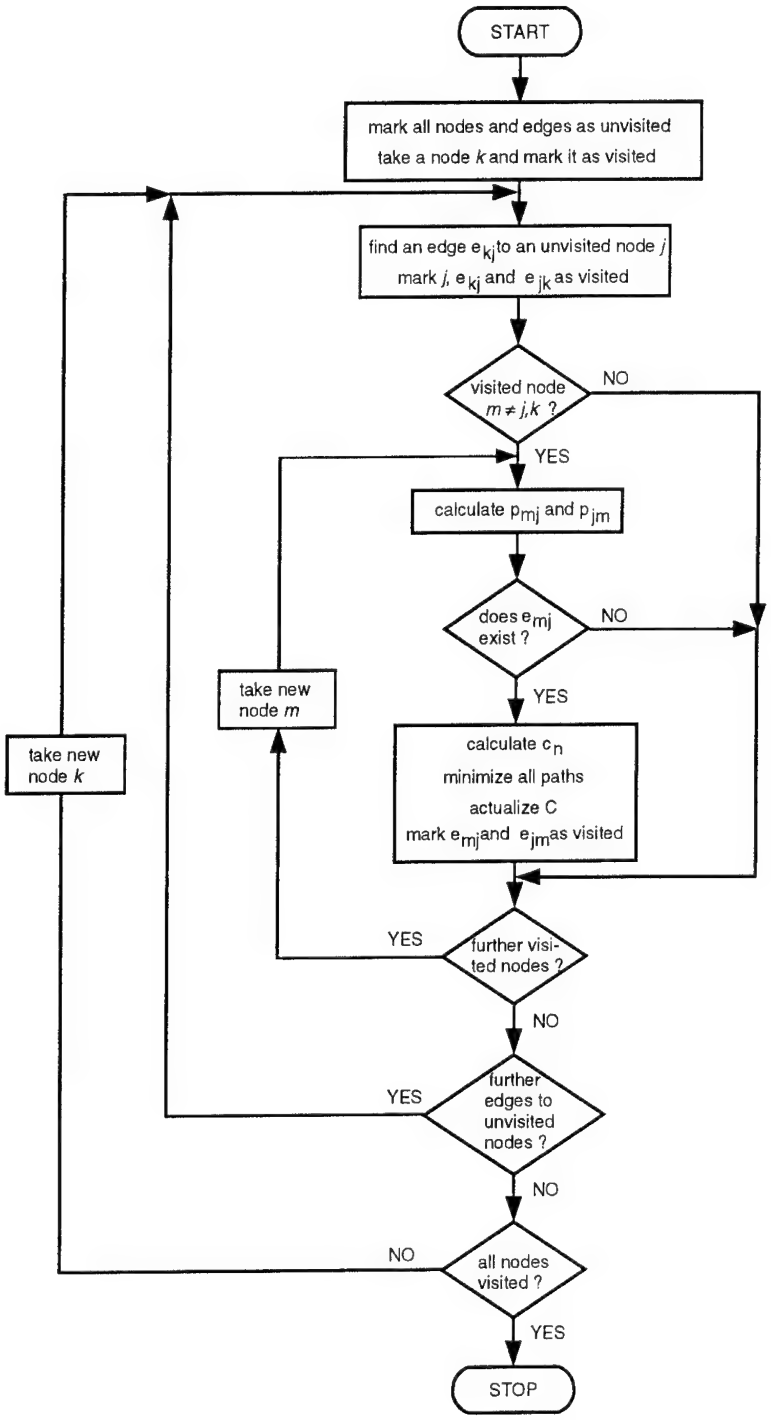



Figure 13: Flow diagram for computation of a minimal basis

of *weighted* edges, in which one associates which each edge a positive number which describes its "length". These lengths can be taken into account when determing the shortest paths, making the algorithm also suitable for finding loops for which the sum of weights is minimal.

## 4.4 Example

In order to illustrate the procedure described above, we apply it to the simple example reproduced in Fig. 14. The graph consists of five nodes and eight edges, which have been weighted by the numbers displayed next to the edges. The node acting as the root is $\mathcal{K}_1$, which is the only one marked "visited" at the start of the procedure. At the outset, the path matrix $P$ is given as:

$$P = \begin{array}{c} \begin{matrix} \mathcal{K}_1 & \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 \end{matrix} \\ \begin{bmatrix} 0 & \{e_{12}\} & \{e_{13}\} & \{e_{14}\} & 0 \\ \{e_{21}\} & 0 & \{e_{23}\} & \{e_{24}\} & 0 \\ \{e_{31}\} & \{e_{32}\} & 0 & 0 & \{e_{35}\} \\ \{e_{41}\} & \{e_{42}\} & 0 & 0 & \{e_{45}\} \\ 0 & 0 & \{e_{53}\} & \{e_{54}\} & 0 \end{bmatrix} \end{array} \begin{matrix} \mathcal{K}_1 \\ \mathcal{K}_2 \\ \mathcal{K}_3 \\ \mathcal{K}_4 \\ \mathcal{K}_5 \end{matrix}$$

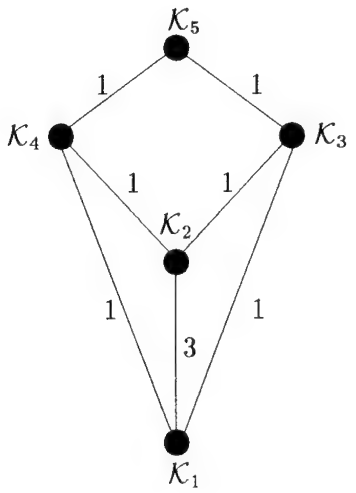



Figure 14: Example for detection of minimal cycle basis

Fig. 15 shows the visited part of the graph, the computed loops, and the new entries of matrix $P$ at the end of each stage. A more detailed reference of the computed shortest paths and smallest cycles is given in Table 2. Stage 1 of the algorithm starts at node $\mathcal{K}_1$, detecting the edge $e_{12}$ to the unvisited node $\mathcal{K}_2$ and marking $\mathcal{K}_2$, $e_{12}$ and $e_{21}$ as visited. As no other visited nodes besides $\mathcal{K}_1$ and $\mathcal{K}_2$ exist at this

point, the algorithm proceeds with the next unvisited edge, which is $e_{13}$. During step 2.1 the shortest path from the visited node $\mathcal{K}_2$ to $\mathcal{K}_3$ is calculated as $p_{23} = e_{21} + e_{13}$, without considering edge $e_{23}$. The path $p_{32}$ is calculated simultanously. The edge $e_{23}$ is recognized in step 2.2, leading to the detection of the first cycle of the cycle basis, which becomes $c_1 = p_{21} \cup \{e_{13}, e_{32}\} = \{e_{21}, e_{13}, e_{32}\}$. The third step at this stage comprises checking if any existing path can be minimized by taking into consideration the new edge $e_{23}$. For example, for the path $p_{23}$, one has three alternatives:

$$\left.\begin{array}{l} p_{23}^{(1)} = \{e_{21}, e_{13}, e_{32}\} \\ p_{23}^{(2)} = \{p_{22}, e_{23}, p_{33}\} = \{e_{23}\} \\ p_{23}^{(3)} = \{p_{23}, e_{32}, e_{32}\} \end{array}\right\} .$$

While the length of $p_{23}^{(2)}$ is one, the length of $p_{23}^{(1)}$ is 4 and the length of $p_{23}^{(3)}$ is 9. Thus the shortest connection between $\mathcal{K}_2$ and $\mathcal{K}_3$ is indeed via the newly added edge $e_{23}$, so the corresponding entries in the path-matrix $P$ are changed accordingly. Similarly, for the path between $\mathcal{K}_1$ and $\mathcal{K}_2$ one has

$$\left.\begin{array}{l} p_{12}^{(1)} = \{e_{12}\} \\ p_{12}^{(2)} = \{e_{12}, e_{23}, e_{32}\} \\ p_{12}^{(2)} = \{e_{13}, e_{32}, p_{22}\} = \{e_{13}, e_{32}\} \end{array}\right\} .$$

Here, the lengths of $p_{12}^{(1)}, p_{12}^{(2)}$ and $p_{12}^{(3)}$ are 3, 5 and 2, respectively. Therefore, one must replace the old shortest path $p_{12}^{(1)}$ by $p_{12}^{(3)}$. The shortest path between the other possible combinations of visited nodes can be calculated in the same way.

Next, the node $\mathcal{K}_4$ is taken as the second unvisited node, while still retaining $\mathcal{K}_1$ as the first node. At this stage, the cycle $c_2 = \{e_{21}, e_{14}, e_{42}\}$ is detected by similar reasonings as those leading to the first cycle. Before adding this cycle to the cycle basis, the algorithm checks whether $c_1^* = c_2 + c_1 = \{e_{13}, e_{32}, e_{24}, e_{41}\}$ is smaller than $c_1$ (step 4). Hereby, $c_1$ has the length 5, while $c_1^*$ has the length 4. Thus the cycle $c_1$ is replaced by the cycle $c_1^*$, and the current cycle basis, at stage 3 of the algorithm, becomes $C = \{c_1^*, c_2\}$.

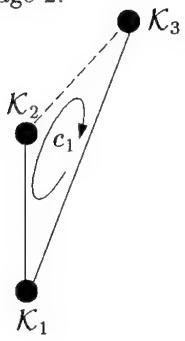
At the beginning of stage 4, no more unvisited nodes remain that are connected to $\mathcal{K}_1$ through an edge. Thus, step 1 of the algorithm is now repeated, taking as next node $\mathcal{K}_3$. From this node, the algorithm detects $\mathcal{K}_5$ as an unvisited node which is connected to $\mathcal{K}_3$. During this stage, $\mathcal{K}_5$ is appended to the visited part of the tree and the cycle basis becomes $C = \{c_1^*, c_2, c_3\}$. After this step all edges and nodes are visited and the algorithm ends.

Stage 1:


$$P = \begin{array}{c} \begin{matrix} \mathcal{K}_1 & \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 \end{matrix} \\ \begin{bmatrix} 0 & \{e_{12}\} & \{e_{13}\} & \{e_{14}\} & 0 \\ \{e_{21}\} & 0 & \{e_{23} & \{e_{24}\} & 0 \\ \{e_{31}\} & \{e_{32}\} & 0 & 0 & \{e_{35}\} \\ \{e_{41}\} & \{e_{42}\} & 0 & 0 & \{e_{45}\} \\ 0 & 0 & \{e_{53}\} & \{e_{54}\} & 0 \end{bmatrix} \end{array} \begin{matrix} \mathcal{K}_1 \\ \mathcal{K}_2 \\ \mathcal{K}_3 \\ \mathcal{K}_4 \\ \mathcal{K}_5 \end{matrix}$$

$$C = \{\,\}$$

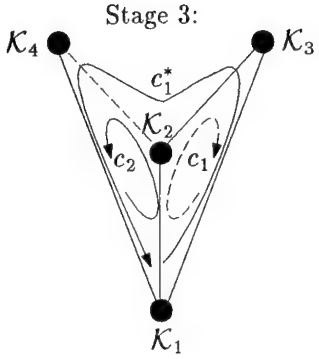
Stage 2:




$$P = \begin{array}{c} \begin{matrix} \mathcal{K}_1 & \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 \end{matrix} \\ \begin{bmatrix} 0 & \{e_{13}, e_{32}\} & \{e_{13}\} & \{e_{14}\} & 0 \\ \{e_{23}, e_{31}\} & 0 & \{e_{23}\} & \{e_{24}\} & 0 \\ \{e_{31}\} & \{e_{32}\} & 0 & 0 & \{e_{35}\} \\ \{e_{41}\} & \{e_{42}\} & 0 & 0 & \{e_{45}\} \\ 0 & 0 & \{e_{53}\} & \{e_{54}\} & 0 \end{bmatrix} \end{array} \begin{matrix} \mathcal{K}_1 \\ \mathcal{K}_2 \\ \mathcal{K}_3 \\ \mathcal{K}_4 \\ \mathcal{K}_5 \end{matrix}$$

$$C = \{c_1\}$$

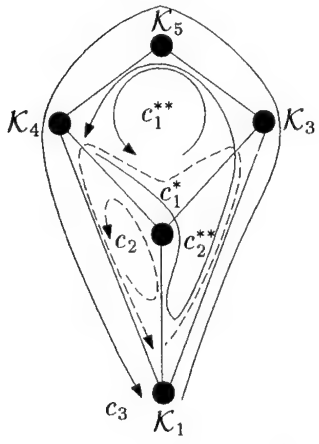
Stage 3:




$$P = \begin{array}{c} \begin{matrix} \mathcal{K}_1 & \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 \end{matrix} \\ \begin{bmatrix} 0 & \{e_{13}, e_{32}\} & \{e_{13}\} & \{e_{14}\} & 0 \\ \{e_{23}, e_{31}\} & 0 & \{e_{23}\} & \{e_{24}\} & 0 \\ \{e_{31}\} & \{e_{32}\} & 0 & \{e_{31}, e_{14}\} & \{e_{35}\} \\ \{e_{41} & \{e_{42}\} & \{e_{41}, e_{13}\} & 0 & \{e_{45}\} \\ 0 & 0 & \{e_{53}\} & \{e_{54}\} & 0 \end{bmatrix} \end{array} \begin{matrix} \mathcal{K}_1 \\ \mathcal{K}_2 \\ \mathcal{K}_3 \\ \mathcal{K}_4 \\ \mathcal{K}_5 \end{matrix}$$

$$C = \{c_1^*, c_2\}$$

Stage 4:




$$P = \begin{array}{c} \begin{matrix} \mathcal{K}_1 & \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 \end{matrix} \\ \begin{bmatrix} 0 & \{e_{13}, e_{32}\} & \{e_{13}\} & \{e_{14}\} & \{e_{13}, e_{35}\} \\ \{e_{23}, e_{31}\} & 0 & \{e_{23}\} & \{e_{24}\} & \{e_{23}, e_{35}\} \\ \{e_{31}\} & \{e_{32}\} & 0 & \{e_{31}, e_{14}\} & \{e_{35}\} \\ \{e_{41}\} & \{e_{42}\} & \{e_{41}, e_{13}\} & 0 & \{e_{45}\} \\ \{e_{53}, e_{31}\} & \{e_{53}, e_{32}\} & \{e_{53}\} & \{e_{54}\} & 0 \end{bmatrix} \end{array} \begin{matrix} \mathcal{K}_1 \\ \mathcal{K}_2 \\ \mathcal{K}_3 \\ \mathcal{K}_4 \\ \mathcal{K}_5 \end{matrix}$$

$$C = \{c_1^*, c_2, c_3\}$$

Figure 15: Processing steps for the minimal-cycle-basis example

| stage | step | $\mathcal{K}_k$ | $\mathcal{K}_j$ | chord | shortest paths | cycle basis |
|---|---|---|---|---|---|---|
| 0 | | | | | $p_{12} = \{e_{12}\}$ $p_{21} = \{e_{21}\}$<br>$p_{13} = \{e_{13}\}$ $p_{31} = \{e_{31}\}$<br>$p_{14} = \{e_{14}\}$ $p_{41} = \{e_{41}\}$<br>$p_{23} = \{e_{23}\}$ $p_{32} = \{e_{32}\}$<br>$p_{24} = \{e_{24}\}$ $p_{42} = \{e_{42}\}$<br>$p_{35} = \{e_{35}\}$ $p_{53} = \{e_{53}\}$<br>$p_{45} = \{e_{45}\}$ $p_{54} = \{e_{54}\}$ | $C = \{\ \}$ |
| 1 | 1 | $\mathcal{K}_1$ | $\mathcal{K}_2$ | | same as in stage 0 | |
| 2 | 1 | $\mathcal{K}_1$ | $\mathcal{K}_3$ | | $p_{23} = \{e_{21}, e_{13}\}$ $p_{32} = \{e_{31}, e_{12}\}$<br>all others same as in stage 1 | |
| | 2 | | | $e_{23}$ | | $c_1 = \{e_{21}, e_{13}, e_{32}\}$<br>$C = \{c_1\}$ |
| | 3 | | | | $p_{23} = \{e_{23}\}$ $p_{32} = \{e_{32}\}$<br>$p_{12} = \{e_{13}, e_{32}\}$ $p_{21} = \{e_{23}, e_{31}\}$<br>all others same as in stage 1 | |
| 3 | 1 | $\mathcal{K}_1$ | $\mathcal{K}_4$ | | $p_{24} = \{e_{21}, e_{14}\}$ $p_{42} = \{e_{41}, e_{12}\}$<br>$p_{34} = \{e_{31}, e_{14}\}$ $p_{43} = \{e_{41}, e_{13}\}$<br>all others same as in stage 2 | |
| | 2 | | | $e_{24}$ | | $c_2 = \{e_{21}, e_{14}, e_{42}\}$ |
| | 3 | | | | $p_{24} = \{e_{21}, e_{14}\}$ $p_{42} = \{e_{41}, e_{12}\}$<br>$p_{34} = \{e_{31}, e_{14}\}$ $p_{43} = \{e_{41}, e_{13}\}$<br>all others same as in stage 2 | |
| | 4 | | | | | $c_1^* = \{e_{13}, e_{32},$<br>$e_{24}, e_{41}\}$<br>$C = \{c_1^*, c_2\}$ |
| 4 | 1 | $\mathcal{K}_3$ | $\mathcal{K}_5$ | | $p_{15} = \{e_{13}, e_{35}\}$ $p_{51} = \{e_{53}, e_{31}\}$<br>$p_{25} = \{e_{23}, e_{35}\}$ $p_{52} = \{e_{53}, e_{32}\}$<br>$p_{45} = \{e_{41}, e_{13}, e_{35}\}$<br>$p_{54} = \{e_{53}, e_{31}, e_{14}\}$<br>all others same as in stage 3 | |
| | 2 | | | $e_{45}$ | | $c_3 = \{e_{41}, e_{13},\}$<br>$e_{13}, e_{35}, e_{54}\}$ |
| | 3 | | | | $p_{15} = \{e_{13}, e_{35}\}$ $p_{51} = \{e_{53}, e_{31}\}$<br>$p_{25} = \{e_{23}, e_{35}\}$ $p_{52} = \{e_{53}, e_{32}\}$<br>$p_{45} = \{e_{45}\}$ $p_{54} = \{e_{54}\}$<br>all others same as in stage 3 | |
| | 4 | | | | | $c_1^{**} = \{e_{32}, e_{24},$<br>$e_{45}, e_{53}\}$<br>$c_2^{**} = \{e_{21}, e_{13}, e_{35},$<br>$e_{54}, e_{41}\}$<br>$C = \{c_1^*, c_2, c_3\}$ |

Table 2: Generated paths and loops for the minimal-cycle-basis example

# 5 Position Analysis of a Single Loop

In the previous two sections it was shown how a general mechanism can be decomposed in a set of clusters, and these, if applicable, in a set of independent loops. The next two sections describe how to produce the associated kinematical equations by regarding first the kinematics of the single loop and then re-assembling these local kinematics to the governing equations for each cluster. The amalgamation of the global kinematics for the set of all clusters is covered in Section **9**.

This section discusses a method for the automated detection and generation of explicit solutions for correspondingly solvable single kinematical loops. Such cases, although not general, occur very often in practical applications, while the general case implies the solution of a polynomial of degree sixteen and thus cannot be solved in closed form.

We regard the kinematical loop as a sequence of homogeneous transformations $\mathrm{A}_i$ , $i = 1, \ldots, n$ , between reference frames $\mathcal{K}_{i-1}$ and $\mathcal{K}_i$ , respectively. The closure of the loop is implied by assuming that the first and last reference frames coincide, i. e., $\mathcal{K}_n \equiv \mathcal{K}_0$ (Fig. 16, Fig. 17). Moreover, we assume that a set of joint variables is contained in the transformations, some of which are independent, and some of which are dependent. Mathematically, the closure condition is formulated as

$$\mathrm{A}_1 \mathrm{A}_2 \cdots \mathrm{A}_n = \boldsymbol{I}_4 \ . \tag{27}$$

Excluding the case of over-constrained mechanisms, the closure condition thus involves twelve non-trivial equations, of which, due to the orthogonality of the rotation part $\boldsymbol{R}$ of the matrix A, only six are independent. These six independent equations can be viewed as implicit equations defining six scalar variables, usually joint coordinates $\beta_1, \ldots, \beta_6$, as a function of the other variables contained in the transformations. The aim of the closed-form solution approach is to derive six scalar equations

$$\left.\begin{aligned} f_1\,(\beta_1) &= 0 \\ f_2\,(\beta_2\ ;\ \beta_1) &= 0 \\ &\vdots \\ f_6\,(\beta_6\ ;\ \beta_5, \ldots, \beta_1) &= 0 \end{aligned}\right\} , \tag{28}$$

with functions $f_i$ containing exactly one unknown more than its preceding equations and being mostly of order two in the corresponding unknown variable $\beta_i$ (or $\tan \frac{\beta_i}{2}$ in the case of a revolute joint). Additionally, the functions $f_i$ should be as simple as possible in order to minimize computational effort.

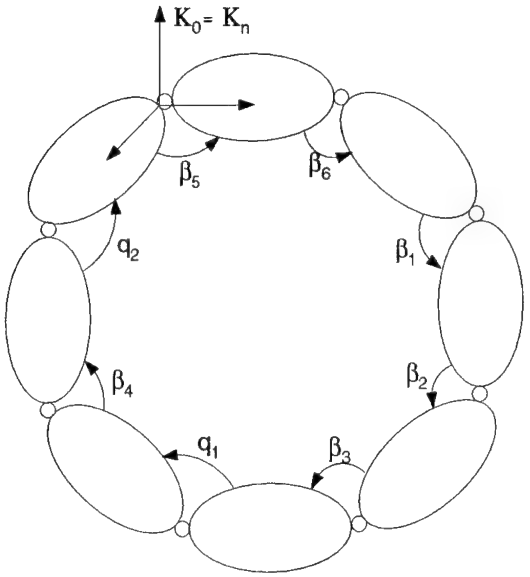



Figure 16: A single kinematical loop

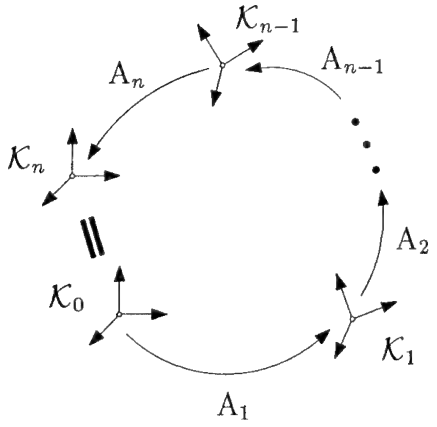



Figure 17: Basic structure of a loop

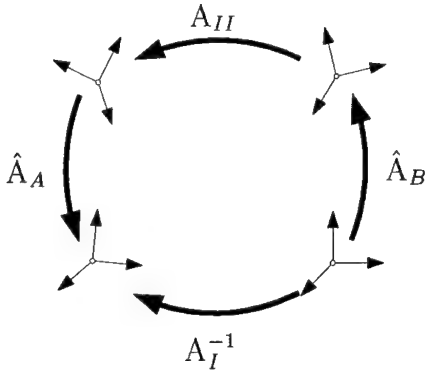



Figure 18: Grouping of transformations

One way of achieving this is to state all possible alternative forms of the closure condition

$$A_{i_{j+1}} \cdots A_{i_k} = A_{i_j}^{-1} \cdots A_{i_1}^{-1} A_{i_n}^{-1} \cdots A_{i_{k+1}}^{-1} \tag{29}$$

where $1 < j < k < n$ and $i_1, \ldots, i_n$ is a cyclic permutation of $1, \ldots, n$, and extract from the resulting set of $12n^2$ equations the ones which are most simple. However, because the number of terms in some of the coefficients of the resulting matrix can grow up to $2^{(n-1)}$, such an approach would not be feasible already for modestly sized loops involving e. g. 10 transformations.

An alternative approach is to group the transformations into four particular sequences $A_I$, $A_{II}$, $\hat{A}_A$ and $\hat{A}_B$, where $\hat{A}_A$ and $\hat{A}_B$ are characterized by the fact

that they are the longest possible sequences of transformations leaving one of the geometric elements *point*, *line*, *plane* or *direction* invariant (Fig. 18). Let $\underline{\xi}_A$ and $\underline{\xi}_B$ denote two such invariant geometric elements. Then, any measurement taken between these two elements will not depend on the transformation sequences $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$ or the unknown variables contained therein, which are thus eliminated. To see this, let the closure condition be restated as

$$\hat{\mathrm{A}}_B \, \mathrm{A}_{II} \, \hat{\mathrm{A}}_A = \mathrm{A}_I^{-1} \quad , \tag{30}$$

and let the measurement be represented as a projection operator $\pi(\underline{\xi}_B, \underline{\xi}_A; \mathrm{A})$ (Fig. 19) where A is the homogeneous transformation describing the motion of the reference system holding $\underline{\xi}_A$ with respect to the reference system holding $\underline{\xi}_B$. Then, after applying the projection operator to both sides of equation (30), one obtains the scalar equation

$$\pi(\underline{\xi}_B, \underline{\xi}_A; \mathrm{A}_{II}) = \pi(\underline{\xi}_B, \underline{\xi}_A; \mathrm{A}_I^{-1}) \tag{31}$$

which does not depend on the unknowns contained in $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$. Currently, five such basic measurements are being used: (I) the quadratic distance between two points $g_{PP}$, (II) the distance of a point to a plane $g_{EP}$, (III) the cosine of the angle between two planes (or orientations) $g_{EE}$, (IV) the distance (along the common perpendicular) between two lines $g_{LL}$ and (V) the quadratic distance of a point to a line $g_{LP}$, see e.g. Hiller and Kecskemethy 1989 (Fig. 20).

## 5.1 Projection Operators

In order to carry out these measurements using homogeneous transformations, two reference frames $\mathcal{K}$ (fixed) and $\mathcal{K}'$ (moved) are introduced, such that the points, lines or planes mentioned above correspond either to the origin $\underline{o}$, a coordinate axis $\mathcal{L}_i$ or a coordinate plane $\Pi_i$, of these frames. Denoting by A the homogeneous matrix relating coordinates of $\mathcal{K}'$ to coordinates of $\mathcal{K}$, where

$$\mathrm{A} = \begin{bmatrix} \boldsymbol{R} & \underline{r} \\ 0 & 1 \end{bmatrix} \tag{32}$$

the measurements mentioned above are carried out by the following operations

$$g_{PP}(A) = \| \mathtt{Trans}[\,\mathrm{A}\,] \|^2 \quad , \tag{33}$$

$$g_{EP}(A; \underline{e}_i) = \overset{\mathrm{H}^{\mathrm{T}}}{\underline{e}_i} \, \mathrm{A} \, \overset{\mathrm{H}}{\underline{o}} \;\; (= r_i(A)) \quad , \tag{34}$$

$$g_{EE}(A; \underline{e}_i, \underline{e}'_j) = \overset{\mathrm{H}^{\mathrm{T}}}{\underline{e}_i} \, \mathrm{A} \, \overset{\mathrm{H}'}{\underline{e}_j} \;\; (= \rho_{ij}(A)) \quad , \tag{35}$$

$$g_{LL}(A; \underline{e}_i, \underline{e}'_j) = \underline{e}_i^{\mathrm{T}} \left[ \mathtt{Rot}[\,\mathrm{A}\,] \, \underline{e}'_j \times \mathtt{Trans}[\,\mathrm{A}\,] \right] \quad , \tag{36}$$

$$g_{LP}(A; \underline{e}_i) = \| \mathtt{Trans}[\,\mathrm{A}\,] \|^2 - (\overset{\mathrm{H}^{\mathrm{T}}}{\underline{e}_i} \, \mathrm{A} \, \overset{\mathrm{H}}{\underline{o}})^2 \quad . \tag{37}$$

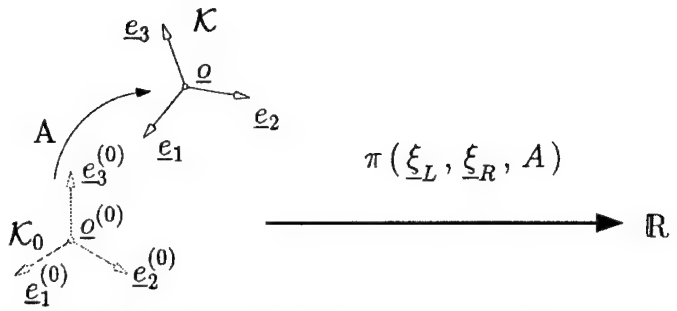



Figure 19: Projection of a spatial transformation to a scalar number

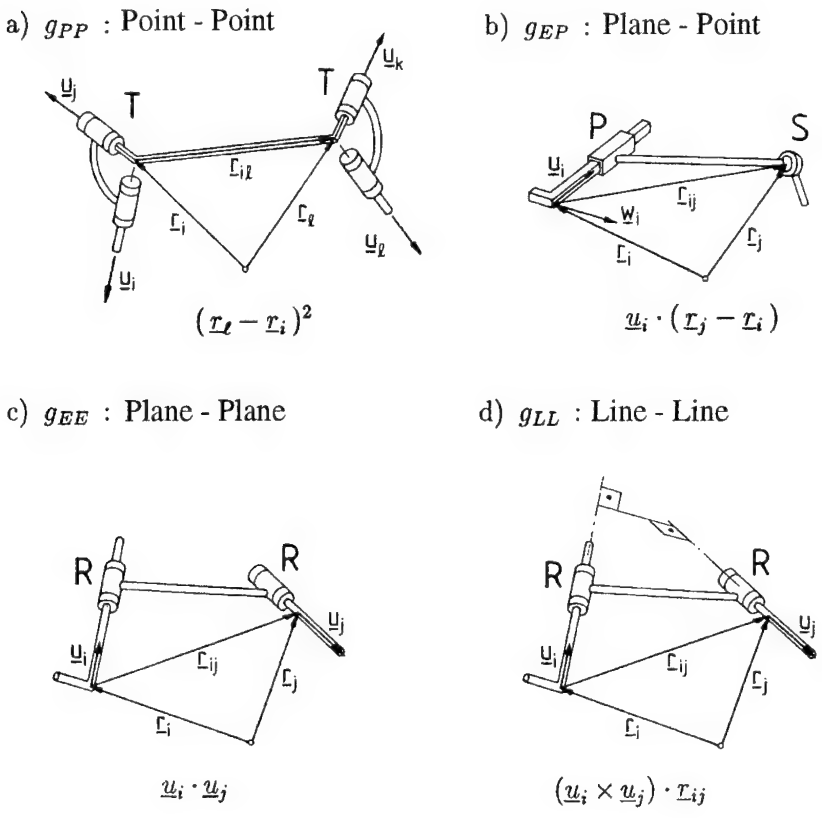



Figure 20: Projection operations

Here, the symbols marked with a stacked "H" denote homogeneous vectors, which can be either points, when the fourth component is 1, or directions, when the fourth component is zero. In particular, one has for the three directions parallel to the coordinate axes and the origin of the coordinate system, respectively,

$$\overset{\mathrm{H}}{\underline{e}}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} , \overset{\mathrm{H}}{\underline{e}}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} , \overset{\mathrm{H}}{\underline{e}}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} , \overset{\mathrm{H}}{\underline{o}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} . \tag{38}$$

Moreover, the following shortcut has been introduced:

$$\mathtt{Rot}[\,\mathrm{A}\,] \equiv \boldsymbol{R} \quad , \quad \mathtt{Trans}[\,\mathrm{A}\,] \equiv \underline{r} \, . \tag{39}$$

Note that for the projections $g_{EP}$ or $g_{LP}$ the corresponding plane or line is chosen from the (fixed) frame $\mathcal{K}$. This is consistent with the property that premultiplication of homogeneous matrices is only meaningful for orientation vectors. In order to resolve the projection for an unknown joint coordinate $\beta$, the matrix A is decomposed as follows

$$\mathrm{A} = \mathrm{A}_\ell \, \mathrm{A}_E(\underline{e}_\nu \, ; \, \beta) \, \mathrm{A}_r = \begin{bmatrix} \boldsymbol{R}_\ell & \underline{r}_\ell \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_E(\underline{e}_\nu \, ; \, \beta) & \underline{r}_E(\underline{e}_\nu \, ; \, \beta) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_r & \underline{r}_r \\ 0 & 1 \end{bmatrix} , \tag{40}$$

where $\beta$ is now contained in the elementary transformation matrix $\mathrm{A}_E(\underline{e}_\nu \, ; \, \beta)$ only. The latter has the structure

$$\boldsymbol{R}_E(\underline{e}_\nu \, ; \, \beta) = \boldsymbol{I}_3 + \overline{\sigma} \left( \sin\beta \, \tilde{\underline{e}}_\nu + (1 - \cos\beta \,) \, \tilde{\underline{e}}_\nu^2 \right) = \boldsymbol{I}_3 + \overline{\sigma} \, T(\underline{e}_\nu \, ; \, \beta) \tag{41}$$

$$\underline{r}_E(\underline{e}_\nu \, ; \, \beta) = \sigma \underline{e}_\nu \, \beta \tag{42}$$

where $\sigma$ is again a Boolean variable indicating the type of the joint ($\sigma = 0$ for revolute joint, $\sigma = 1$ for prismatic joint) and the tilde denotes the operation of generating an anti-symmetric matrix from an euclidean vector as

$$\tilde{\underline{v}} = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix} .$$

After carrying out all multiplications, the following structure is obtained for projection types $g_{PP}$ , $g_{EP}$ , $g_{EE}$ , $g_{LL}$

$$g_{(\ldots)}(\mathrm{A}(\beta)) = \begin{cases} A \cos\Theta + B \sin\Theta + C & \text{for} \quad \sigma = 0 \quad (\beta \equiv \Theta) \\ A \, s^2 + B \, s + C & \text{for} \quad \sigma = 1 \quad (\beta \equiv s) \end{cases} , \tag{43}$$

while projection type $g_{LP}$ gives

$$g_{LP}(\mathrm{A}(\beta)) = \begin{cases} A^* \cos^2\Theta + B^* \cos\Theta \sin\Theta + A\cos\Theta + B\sin\Theta + C & \text{for } \sigma = 0 \\ A\, s^2 + B\, s + C & \text{for } \sigma = 1 \end{cases} . \quad (44)$$

The projections $g_{PP}$ , $g_{EP}$ , $g_{EE}$ , $g_{LL}$ can be used for easy resolution both for angular and translational unknowns, while the projection $g_{LP}$ yields simple results only in the case of a translational unknown. Also, for translational unknowns, the projection type $g_{EE}$ gives always vanishing coefficients $A, B, C$ , while the projections $g_{EP}$ and $g_{LL}$ yield vanishing coefficient $A$ . For computation of explicit solutions the three projections $g_{PP}$ , $g_{EP}$ , $g_{EE}$ are the most important ones, because equation (44) can be solved explicitly only for translational unknowns and projection $g_{LL}$ only occurs in combination with transformation sequences $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$ which include less then five of the six possible unknowns (Woernle 1988).

Projections $g_{EP}$ and $g_{EE}$ have an interesting property when the unknown is an angular variable $\Theta$ . If the transformation $\mathrm{A}_\ell$ in equation (40) is the identity, one can repeat the projection operation after exchanging the original left vector $\underline{e}_i$ with $\underline{e}_\nu \times \underline{e}_i$, where $\underline{e}_\nu \times \underline{e}_i$ is the axis of action of the elementary transformation and $\nu \neq i$ . Then, this second projection yields a function $g^{(2)}_{(\ldots)}(\Theta)$ which is related to the original projection, $g^{(1)}_{(\ldots)}(\Theta)$ , by

$$\left.\begin{array}{rcl} g^{(1)}_{(\ldots)}(\Theta) & = & A\cos\Theta + B\sin\Theta + C^{(1)} \\ g^{(2)}_{(\ldots)}(\Theta) & = & -B\cos\Theta + A\sin\Theta + C^{(2)} \end{array}\right\} . \quad (45)$$

From this set of equations, the angle $\Theta$ can be determined uniquely in the range $(-\pi, \pi]$ while for equation (43) two solutions can be found:

$$\beta = \begin{cases} \tan^{-1}(\frac{A}{B}) \pm \tan^{-1}\sqrt{\frac{A^2+B^2}{C^2} - 1} & \text{for } \sigma = 0 \quad (\beta \equiv \Theta) \\ -\frac{B}{2A} \pm \frac{\sqrt{B^2 - 4AC}}{2A} & \text{for } \sigma = 1 \quad (\beta \equiv s) \end{cases} , \quad (46)$$

## 5.2 Possible Structures of the Constraint Equations

From the discussions above, it is clear that in any multibody loop one can always find two geometric elements, $\underline{\xi}_A$ and $\underline{\xi}_B$, of the type point, line or plane, for which there exist corresponding transformation sequences $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$ that leave these geometric elements invariant. Moreover, for each pair of geometric elements there is an associated projection operator which, when applied to both sides of Eq. (30), produces a scalar equation in which the effect of the transformation sequences $\hat{\mathrm{A}}_A$

and $\hat{\mathrm{A}}_B$, and in particular the unknowns contained therein, can be eliminated. Thus, if for a given loop one can find two particular geometric elements $\underline{\xi}_A$ and $\underline{\xi}_B$ and associated transformation sequences $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$, such that the latter contain five unknowns, the remaining scalar equation will depend only on the remaining one unknown, and thus will be solvable in closed form. Furthermore, after resolving such an equation, all further unknowns can be established in closed form. Thus, these cases give rise to what we term "explicitly solvable loops".

If the transformation sequences $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$ contain less than five unknowns, the closure of the loop will be representable as a polynomial of degree four to sixteen (Raghavan and Roth 1993). Such cases are most efficiently solved by iterative algorithms (see Section **8**), and are thus not treated in this setting.

After finding a first closure condition that can be resolved for one unknown, the remaining unknowns can be determined by repeating the search of invariant geometric elements until no unknowns remain. In this way, an iterative algorithm results that can be applied repeatedly to the loop until explicitly resolvable equations are produced for all unknowns. By a slight modification of the algorithm, it is possible also to cover some overconstrained mechanisms which are *not paradoxical* in the sense defined in Angleles 1988. Hereby, one considers the following five cases

$$\hat{\mathrm{A}}_B\,\mathrm{A}_{II}\,\hat{\mathrm{A}}_A\,\mathrm{A}_I = \boldsymbol{I}_4 \qquad (\underline{\text{Closure Type 0}}) \tag{47}$$

$$\hat{\mathrm{A}}_A = \boldsymbol{I}_4 \qquad (\underline{\text{Closure Type 1}}) \tag{48}$$

$$\hat{\mathrm{A}}_B\,\hat{\mathrm{A}}_A = \boldsymbol{I}_4 \qquad (\underline{\text{Closure Type 2}}) \tag{49}$$

$$\mathrm{A}_I\,\overline{\mathrm{A}}_B\,\mathrm{A}_{II}\,\hat{\mathrm{A}}_A = \boldsymbol{I}_4 \qquad (\underline{\text{Closure Type 3}}) \tag{50}$$

$$\mathrm{A}_I\,\overline{\mathrm{A}}_A = \boldsymbol{I}_4 \qquad (\underline{\text{Closure Type 4}}) \tag{51}$$

The closure equation of Type 0 corresponds to the case where two transformation sequences are found that leave, respectively, one geometric elements invariant, as described above. A closure equation of Type 1, if it arises, means that the whole sequence of transformations in the loop leaves some geometric element $\underline{\xi}_A$ invariant. Thus, any projection carried out with this element yields a scalar equation in which all transformations are eliminated, i. e., that is identically fulfilled. As there are three independent projections which can be carried out with one geometric element, this closure condition produces three identically fulfilled independent constraint equations, reducing the number of relevant constraint equations by three. For example, in the case of a planar four-bar mechanism, all transformations share as invariant element the plane perpendicular to the rotation axes, which produces three identically fulfilled constraint equations. Thus, for the planar mechanism only three non-trivial constraint equations remain for the unknowns in the loop. A similar effect is represented by the closure condition of Type 2. Here, the chain is decomposed into two sequences of transformations that leave, respectively, the geometric element $\underline{\xi}_A$ and $\underline{\xi}_B$ invariant. Then, the associated projection operator produces a scalar equation

which is independent of all unknowns of the loop, and is thus fulfilled identically. Such a situation arises for example in the case of a Cardan shaft, where the six rotational joints can be grouped into two sets of three intersecting axes, with the intersection points corresponding to the respective invariant geometric elements. Note that, for the last two types of closure conditions, the sequences $\hat{\mathrm{A}}_A$ or $\hat{\mathrm{A}}_B$ may be bordered with additional transformations containing no unknowns, without changing the basic results. Such bordering transformations have been intentionally left out for better clarity.

The closure conditions of Type 3 and Type 4 do not occur in the initial stage of the analysis of the loop. Instead, they appear in the subsequent stages in which, by repetition of the searching algorithm, the remaining unknowns contained in the sequences $\hat{\mathrm{A}}_B$ and $\hat{\mathrm{A}}_A$ are determined. This repeated application of the algorithm is as follows: after having produced the projection pertaining to $\underline{\xi}_B$ and $\underline{\xi}_A$, the invariance property associated with $\underline{\xi}_B$ is removed from the elements of $\hat{\mathrm{A}}_B$ together with the current resolved unknown. Then, a new closure condition is searched for by applying the same criteria as above. Eventually, no more invariant properties remain besides those in $\hat{\mathrm{A}}_A$, but there is still an unknown in the remaining transformations. This is the situation in the closure condition of Type 3, where $\overline{\mathrm{A}}_B$ contains the remaining unknown in a form similar to equation (40). Then, a projection as defined in equation (31) is carried out, but this time $\underline{\xi}_B$ is taken as a geometric element which actually *is* transformed by $\overline{\mathrm{A}}_B$, thus yielding a scalar equation which contains this unknown. In the case that $\overline{\mathrm{A}}_B$ is a rotation, there are two geometric elements that are transformed, namely, the coordinate planes parallel to the rotational axis. By taking these as projection elements, a pair of uniquely solvable equations is obtained. For the case of a prismatic joint, the geometric elements which are not invariant are the origin of the reference frame and the plane normal to the joint axis. After performing this step, the invariance properties of $\underline{\xi}_A$ are removed, and the process described above is repeated. Eventually, one only one unknown remains, while all invariant properties have been eliminated. Then, a closure condition of Type 4 arises, where $\overline{\mathrm{A}}_A$ contains the remaining unknown. After projecting as stated in Eq. (31), with $\underline{\xi}_B$ and $\underline{\xi}_A$ taken as geometrical elements which are actually transformed by $\overline{\mathrm{A}}_A$, a unique solution as in the case of the closure condition of Type 3 is obtained.

## 5.3 Implementation of the Algorithm

Below we describe an implementation of the concepts described above. As geometric elements, the origin $o$ and the three planes $\Pi_1, \Pi_2, \Pi_3$ normal to the coordinate axes $x, y, z$, respectively, and passing through the origin are regarded. The generalisation of the implementation so that lines are also considered is straight-forward and shall not be discussed here. For the practical implementation of the algorithm, a special matrix is constructed which is denominated the *invariance properties matrix* IPM.

In this matrix, each transformation $A_i$ in the chain is assigned a column with elements $[\sigma_{\Pi_1}, \sigma_{\Pi_2}, \sigma_{\Pi_3}, \sigma_o]^T$ that are equal to one if the corresponding geometric element is invariant under the transformation $A_i$ and zero otherwise. Sequences that leave a geometric element invariant can be recognized as sets of adjacent elements exhibiting a pattern of contiguous ones in the same row. After marking columns containing unknowns, sequences $\hat{A}_A$ and $\hat{A}_B$ can be recognized as the two non-overlapping sets of contiguous ones which contain the maximum number of marked columns. $\hat{A}_A$ is determined first, whereby at this stage the first and last columns of IPM can be regarded as neighbours, as the order of transformations can be cyclically interchanged. Then, $\hat{A}_B$ is determined from the matrix IPM′ resulting from IPM after removing the columns corresponding to $\hat{A}_A$.

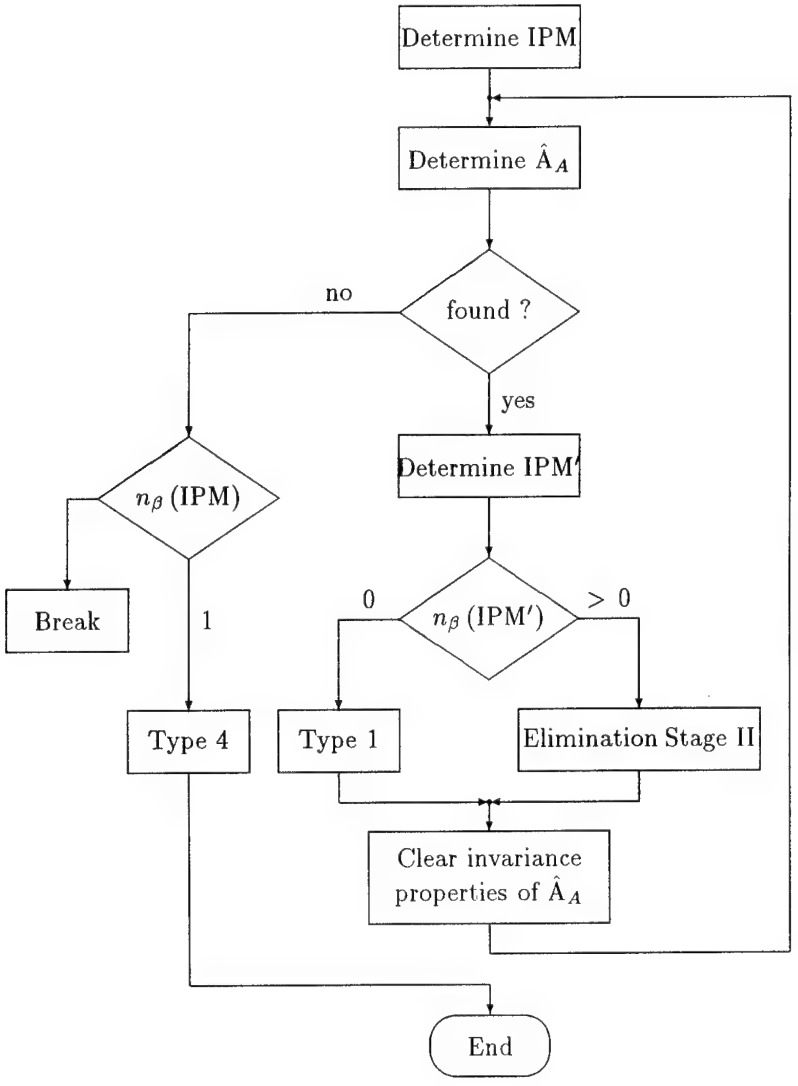



Figure 21: Flow diagram for stage I of single-loop processing algorithm

Fig. 21 and Fig. 22 show the flow diagram of the algorithm. One can clearly see the

simple branching conditions which lead to the different closure conditions Type 0 through Type 4 discussed above. In this flow diagrams, the symbol $n_\beta(\mathrm{A})$ stands for the number of unknowns remaining after performing the $i$-th projection. Also, all stages of the equation generation can be processed by the same piece of code, which encompasses only about 1600 lines of *Mathematica* code. The implementation is organized in four stages: (1) processing of direct kinematics, (2) generation of invariance properties matrices, (3) evaluation of projections, and (4) control of equation selection. An advantage of the code is the simple input format for a mechanism, which can be defined using the well-established Denavit-Hartenberg parametrisation. This is shown below for some examples.

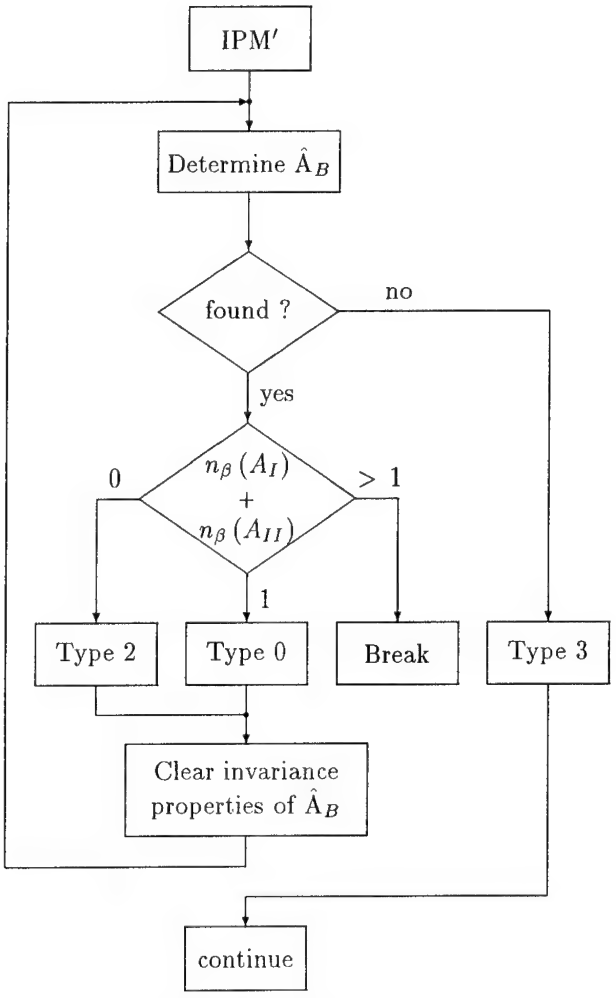
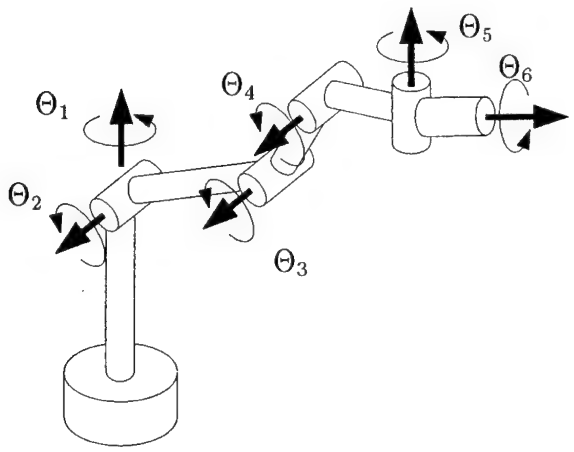



Figure 22: Flow diagram for stage II of single-loop processing algorithm

## 5.4 Examples

As a first example, consider the elbow manipulator depicted in Fig. 23, which is also treated in Hunt 1986. The Denavit-Hartenberg parameters of the robot are as given in the table of Fig. 23.




| $\Theta$ | $d$ | $\alpha$ | $a$ |
|---|---|---|---|
| $\Theta_1$ | 0 | $-\pi/2$ | 0 |
| $\Theta_2$ | 0 | 0 | $f$ |
| $\Theta_3$ | 0 | 0 | $g$ |
| $\Theta_4$ | 0 | $\pi/2$ | $h$ |
| $\Theta_5$ | 0 | $-\pi/2$ | 0 |
| 0 | 0 | $\Theta_6$ | 0 |

Figure 23: An elbow manipulator and its Denavit-Hartenberg parameters

The input for the program consists of a list containing a transformation sequence and a list of unknowns in the form {`transformation.transformation...,{vars}`}.

```
In[1]:= ElbowManipulator =
                                -Pi
        {DHTransform[theta1, 0, ---, 0] . DHTransform[theta2, 0, 0, f] .
                                 2                                  Pi
>        DHTransform[theta3, 0, 0, g] . DHTransform[theta4, 0, --, h] .
                                -Pi                                 2
>        DHTransform[theta5, 0, ---, 0] . DHTransform[0, 0, theta6, 0] .
                                 2
>        Inverse[ATransform[{{nL, n[1, 2], n[1, 3]}, {nM, n[2, 2], n[2, 3]},
>           {nN, n[3, 2], n[3, 3]}}, {x05, y05, z05}]],
>         {theta1, theta2, theta3, theta4, theta5, theta6}};

In[2]:= SimplifyChain[%]
Out[2]= ETransform[3, 0, theta1] . ETransform[2, 0, theta2] .
>       ETransform[1, 1, f] . ETransform[2, 0, theta3] . ETransform[1, 1, g] .
>       ETransform[2, 0, theta4] . ETransform[1, 1, h] .
                                                      -Pi
>       ETransform[3, 0, theta5] . ETransform[1, 0, --- + theta6] .
                                                       2
>       Inverse[ATransform[{{nL, n[1, 2], n[1, 3]}, {nM, n[2, 2], n[2, 3]},
>          {nN, n[3, 2], n[3, 3]}}, {x05, y05, z05}]]
```

For the example at hand, the transformations are defined either as (1) Denavit-Hartenberg expressions `DHTransform`[theta,d,alpha,a], in which the parameters correspond to the ones in the expression defined in Table 1, (2) as a general transformation `ATransform`[...] consisting of a rotation matrix defined in row-wise format and the radius vector, or (3) as elementary transformations `ETransform`$[k, \sigma, \beta]$. The unknowns comprise the variables `theta1,...,theta6`. Note that the closure of the chain is defined as the inverse of the prescribed motion of the end-effector. `Out[2]` shows the reduction of the user-supplied chain of transformations to the normal form comprising only general and elementary transformations. Clearly, there are ten such transformations, which shall be denoted by $\mathrm{A}_1, \ldots, \mathrm{A}_{10}$.

Next, the *invariance properties matrix* is generated and further processed. The initial values of the elements of this matrix are

$$\mathrm{IPM} = \begin{array}{c|cccccccccc|l} & \mathrm{A}_1 & \mathrm{A}_2 & \mathrm{A}_3 & \mathrm{A}_4 & \mathrm{A}_5 & \mathrm{A}_6 & \mathrm{A}_7 & \mathrm{A}_8 & \mathrm{A}_9 & \mathrm{A}_{10} & \\ \hline & & & & & & & & & 1 & & \Pi_1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 & & & & \Pi_2 \\ & 1 & & 1 & & 1 & & 1 & 1 & & & \Pi_3 \\ & 1 & 1 & & 1 & & 1 & & 1 & 1 & & \underline{o} \\ \hline & \Theta_1 & \Theta_2 & & \Theta_3 & & \Theta_4 & & \Theta_5 & \Theta_6 & & \end{array} , \tag{52}$$

where the zeros have been omitted for better clarity. For example, column 4 states that the fourth transformation, which is a rotation about the $y$-axis, leaves the plane normal to the $y$-axis and the origin of the local reference frame invariant. The further values of the elements during the different steps of the algorithm are displayed in Fig. 24. With the invariance properties matrix IPM displayed above, the sequence $\hat{\mathrm{A}}_A$ can be easily detected in row 2, where it comprises columns 2–7 of adjacent ones, and covers three 'marked' columns with unknowns $\Theta_2, \Theta_3, \Theta_4$. Similarly, $\hat{\mathrm{A}}_B$ is detected in row 4 by the two adjacent ones in columns 8 and 9, with unknowns $\Theta_5$ and $\Theta_6$. Thus the optimized closure condition is:

$$\overbrace{\mathrm{A}_8\,\mathrm{A}_9}^{\hat{\mathrm{A}}_B}\;\overbrace{\mathrm{A}_{10}\,\mathrm{A}_1}^{\mathrm{A}_{II}}\;\overbrace{\mathrm{A}_2\,\mathrm{A}_3\,\mathrm{A}_4\,\mathrm{A}_5\,\mathrm{A}_6\,\mathrm{A}_7}^{\hat{\mathrm{A}}_A} = \boldsymbol{I}_4 \; .$$

Note that here the columns 1 and 10 are viewed as contiguous, as in first step of the algorithm all transformations can be cyclically permuted.

After applying the general projection operator with $\underline{\xi}_A = \Pi_2$ and $\underline{\xi}_B = \underline{o}$, the transformations contained in $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$ are eliminated and a scalar equation is obtained in which the only remaining unknown is $\Theta_1$. After this, the invariance properties of $\hat{\mathrm{A}}_B$ are temporarily removed, i. e. the ones in row 4, columns 8–9, of matrix IPM are ignored.

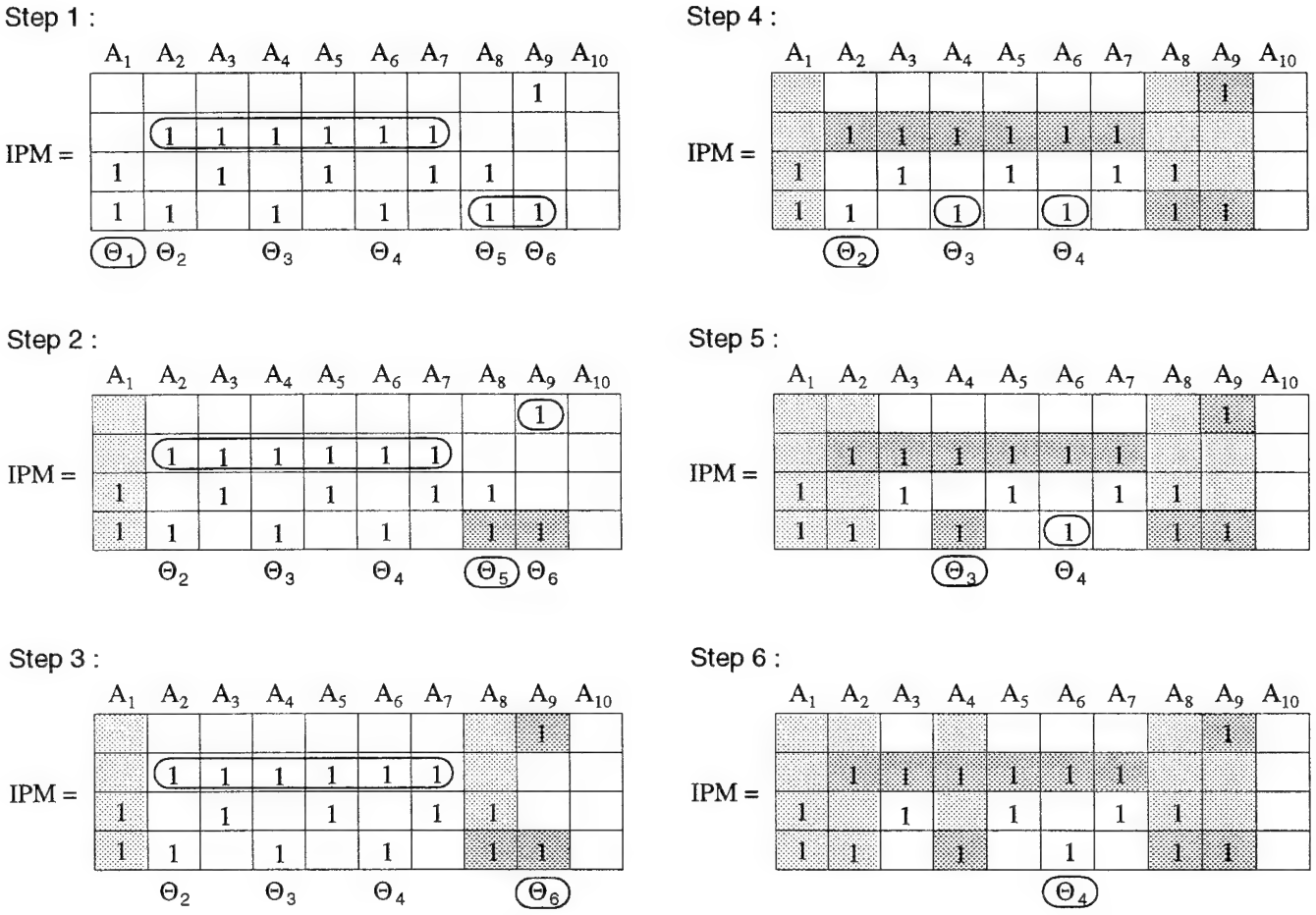



Figure 24: IPM-matrices for the elbow manipulator

The next set of transformations featuring invariance properties and containing unknowns is detected in column 9, row 1, stating that the plane normal to the $x$-axis of the local reference frame is invariant with respect to transformations by $\Theta_6$. Applying the projection with the elements $\Pi_2$ and $\Pi_1$ thus generates a scalar equation for the remaining unknown $\Theta_5$. After removing the corresponding invariance properties from IPM, no further invariance properties remain besides those contained in $\hat{\mathrm{A}}_A$. However, one unknown, namely $\Theta_6$, still remains in column 9. Thus, the situation of closure Type 4 arises, and one has to carry out the projections with the *non*-invariant elements of $\mathrm{A}_9$, which are $\Pi_2$ and $\Pi_3$. In this way, a pair of constraint equations as described in Eq. (45) is obtained. After this step, neither unknowns nor invariance properties remain in IPM, so the invariance properties of $\hat{\mathrm{A}}_A$ are removed by cancelling the ones in row 2, columns 2–7. Then, the whole process is started over again, but now without allowing for cyclic permutations. The resolvable unknowns, decompositions and geometric elements resulting from these steps, as well as the ones described above, are

$$\Theta_5 : \quad \overbrace{\mathrm{A}_8}^{\mathrm{A}_I}\,\overbrace{\mathrm{A}_9}^{\hat{\mathrm{A}}_B}\,\overbrace{\mathrm{A}_{10}\,\mathrm{A}_1}^{\mathrm{A}_{II}}\,\overbrace{\mathrm{A}_2\,\mathrm{A}_3\,\mathrm{A}_4\,\mathrm{A}_5\,\mathrm{A}_6\,\mathrm{A}_7}^{\hat{\mathrm{A}}_A} = \boldsymbol{I}_4 \;;\; \underline{\xi}_A = \Pi_2\,,\, \underline{\xi}_B = \Pi_1$$

$$\Theta_6 : \quad \overbrace{\mathrm{A}_8}^{\mathrm{A}_I}\,\overbrace{\mathrm{A}_9}^{\overline{\mathrm{A}}_B}\,\overbrace{\mathrm{A}_{10}\,\mathrm{A}_1}^{\mathrm{A}_{II}}\,\overbrace{\mathrm{A}_2\,\mathrm{A}_3\,\mathrm{A}_4\,\mathrm{A}_5\,\mathrm{A}_6\,\mathrm{A}_7}^{\hat{\mathrm{A}}_A} = \boldsymbol{I}_4 \;;\; \underline{\xi}_A = \Pi_2\,,\, \underline{\xi}_B = \Pi_2 \text{ and } \Pi_3$$

$$\Theta_2 : \quad \overbrace{\underbrace{\mathrm{A}_7\,\mathrm{A}_8\,\mathrm{A}_9\,\mathrm{A}_{10}\,\mathrm{A}_1}_{\mathrm{A}_l}\,\mathrm{A}_2(\Theta_2)\,\underbrace{\mathrm{A}_3}_{\mathrm{A}_r}}^{\mathrm{A}_I}\,\overbrace{\mathrm{A}_4}^{\hat{\mathrm{A}}_B}\,\overbrace{\mathrm{A}_5}^{\mathrm{A}_{II}}\,\overbrace{\mathrm{A}_6}^{\hat{\mathrm{A}}_A} = \boldsymbol{I}_4 \;;\; \underline{\xi}_A = \underline{o}\;\,,\, \underline{\xi}_B = \underline{o}$$

$$\Theta_3 : \quad \overbrace{\mathrm{A}_7\,\mathrm{A}_8\,\mathrm{A}_9\,\mathrm{A}_{10}\,\mathrm{A}_1\,\mathrm{A}_2\,\mathrm{A}_3}^{\mathrm{A}_I}\,\overbrace{\mathrm{A}_4}^{\overline{\mathrm{A}}_B}\,\overbrace{\mathrm{A}_5}^{\mathrm{A}_{II}}\,\overbrace{\mathrm{A}_6}^{\hat{\mathrm{A}}_A} = \boldsymbol{I}_4 \;;\; \underline{\xi}_A = \underline{o}\;\,,\, \underline{\xi}_B = \Pi_1 \text{ and } \Pi_3$$

$$\Theta_4 : \quad \overbrace{\mathrm{A}_7\,\mathrm{A}_8\,\mathrm{A}_9\,\mathrm{A}_{10}\,\mathrm{A}_1\,\mathrm{A}_2\,\mathrm{A}_3\,\mathrm{A}_4\,\mathrm{A}_5}^{\mathrm{A}_I}\,\overbrace{\mathrm{A}_6}^{\overline{\mathrm{A}}_A} = \boldsymbol{I}_4 \;;\; \underline{\xi}_A = \underline{\xi}_B = \Pi_1 \text{ and } \Pi_3$$

Knowing that the projected equation has either the structure of Eq. (43) or Eq. (45), the code just needs to generate the coefficients $A, B, C$ in the first case and $A, B, C^{(1)}, C^{(2)}$ in the second case. This is done through constructs termed *Explicit Solution Forms*, printed as $\mathtt{ESF}[\sigma, \Theta_i, A, B, C]$ or $\mathtt{ESF}[\sigma, \Theta_i, A, B, C^{(1)}, C^{(2)}]$, which specify the coefficients of the scalar equation, the unknown, as well as the type of the unknown, which is angular for $\sigma = 0$ and translational for $\sigma = 1$. The explicit scalar equations produced by the code are:

```
In[3]:= GenerateConstraints[ElbowManipulator]
Out[3]= {ESF[0, -theta1, y05, x05, 0], ESF[0, -theta5, 0, -1, -ADH1[1, 2]],
                 -Pi
>        ESF[0, --- + theta6, ADH1[2, 2], -ADH1[3, 2], -Cos[-theta5], 0],
                  2
>        ESF[0, -theta2, -2 f ADH3[1, 4], -2 f ADH4[3, 4],
              2    2             2             2             2
>           f  - g  + ADH3[1, 4]  + ADH3[2, 4]  + ADH4[3, 4] ],
```

```
>       ESF[0, theta3, 2 f g, 0,

            2    2            2            2            2
           f  + g  - ADH4[1, 4]  - ADH4[2, 4]  -  ADH4[3, 4] ],
>       ESF[0, theta2 + theta3 + theta4, 0, 1, -ADH3[1, 3], -ADH4[3, 3]]}

In[4]:= GetSubstitutions[%]
Out[4]= {ADH1[1, 2] -> nM Cos[theta1] - nL Sin[theta1],
>       ADH1[2, 2] -> -(Sin[theta1] n[1, 2]) + Cos[theta1] n[2, 2],
>       ADH1[3, 2] -> -(Sin[theta1] n[1, 3]) + Cos[theta1] n[2, 3],
>       ADH3[1, 3] -> ADH4[1, 3] Cos[theta1] + ADH4[2, 3] Sin[theta1],
>       ADH3[1, 4] -> ADH4[1, 4] Cos[theta1] + ADH4[2, 4] Sin[theta1],
>       ADH3[2, 4] -> ADH4[2, 4] Cos[theta1] - ADH4[1, 4] Sin[theta1],
>       ADH4[1, 4] -> x05 + nL ADH6[1, 4] + ADH5[2, 4] n[1, 2] +
                      ADH5[3, 4] n[1, 3],
>       ADH4[2, 4] -> y05 + nM ADH6[1, 4] + ADH5[2, 4] n[2, 2] +
                      ADH5[3, 4] n[2, 3],
>       ADH4[3, 3] -> -(Cos[theta6] n[3, 2]) + Sin[theta6] n[3, 3],
>       ADH4[3, 4] -> z05 + nN ADH6[1, 4] + ADH5[2, 4] n[3, 2] +
                      ADH5[3, 4] n[3, 3],
>       ADH4[1, 3] -> -(Cos[theta6] n[1, 2]) + Sin[theta6] n[1, 3],
>       ADH4[2, 3] -> -(Cos[theta6] n[2, 2]) + Sin[theta6] n[2, 3],
>       ADH5[2, 4] -> ADH6[2, 4] Sin[theta6],
>       ADH5[3, 4] -> ADH6[2, 4] Cos[theta6],
>       ADH6[1, 4] -> -(h Cos[theta5]),
>       ADH6[2, 4] -> h Sin[theta5]}
```

`Out[4]` shows the substitutions introduced automatically to reduce computational overhead. These terms are shortcuts for lengthy expressions and are denoted by `ADH`$k$`[`$i$`,`$j$`]`, as they are typically elements of $4 \times 4$ homogenous transformation matrices. Hereby, $i$ and $j$ denote the row and column indices of the matrix elements, while $k$ is a running index for an intermediate matrix. It must be noted that the resulting expressions for $\Theta_3$ and $\Theta_4$ have been obtained using slight modifications of the procedure described above, which take into account parallel axes. The results coincide almost completely with those reported in Hunt 1986.

An example of an over-constrained mechanism for which explicit solutions can be generated is the trailer wheel suspension described in section 2.6.2. The system is planar, all rotational axes being parallel to the global $x$-axis. Supposing that the length $s$ of the shock absorber is taken as independent coordinate, and that the three angles $\alpha$, $\beta$ and $\gamma$ are taken as the unknowns, the program detects that $\Pi_1$ is invariant under all transformations, and thus generates three trivial equations containing the spurious 'variable' `Null`. These trivial equations printed together with the relevant solutions for the remaining three dependent joint coordinates make the six scalar equations for the loop, which become

```
    ESF[1, Null, 0, 0, 0],
```

```
ESF[1, Null, 0, 0, 0],
ESF[1, -Null, 0, 1, 0],
ESF[0, beta, -2 s v2z, -2 s ADH1[2, 4],
>    ADH2[1, 1] + ADH3[1, 1] - ADH5[1, 1]],
ESF[0, alpha, -v3y, -v3z, -ADH6[2, 4], -ADH7[3, 4]],
ESF[0, -gamma, 1, 0, -ADH10[2, 2], -ADH10[3, 2]]
```

The explicit values of the expressions for the shortcuts are obtained by issuing the command `SolvePositionEquations[%]`, yielding

```
ADH1[2, 4] -> -v1y + v2y
                2
ADH3[1, 1] -> s
                  2      2
ADH5[1, 1] -> v3y  + v3z
                  2             2
ADH2[1, 1] -> v2z  + ADH1[2, 4]
                                    2     2             2
                                 4 s  (v2z  + ADH1[2, 4] )
beta -> -ArcTan[Sqrt[-1 + ----------------------------------------]] +
                                                                  2
                          (ADH2[1, 1] + ADH3[1, 1] - ADH5[1, 1])
>    ArcTan[-2 s v2z, -2 s ADH1[2, 4]]
ADH8[2, 4] -> s Sin[beta]
ADH8[3, 4] -> -(s Cos[beta])
ADH6[2, 4] -> -v1y + v2y + ADH8[2, 4]
ADH7[3, 4] -> -v2z + ADH8[3, 4]
                -(v3y ADH6[2, 4]) + v3z ADH7[3, 4]
alpha -> ArcTan[--------------------------------,
                             2      2
                          v3y  + v3z
       v3z ADH6[2, 4] + v3y ADH7[3, 4]
>    -(------------------------------)]
                   2      2
                v3y  + v3z
ADH10[2, 2] -> Cos[alpha] Cos[beta] + Sin[alpha] Sin[beta]
ADH10[3, 2] -> Cos[beta] Sin[alpha] - Cos[alpha] Sin[beta]
gamma -> ArcTan[ADH10[2, 2], -ADH10[3, 2]]
```

Note that $\beta$ is computed by solving an equation of type (43) for which two solutions can be found. The above solution fo $\beta$ is thus just one possible solution . A second solution for the variable would be

```
                                    2     2             2
                                 4 s  (v2z  + ADH1[2, 4] )
beta -> +ArcTan[Sqrt[-1 + ----------------------------------------]] +
                                                                  2
                          (ADH2[1, 1] + ADH3[1, 1] - ADH5[1, 1])
>    ArcTan[-2 s v2z, -2 s ADH1[2, 4]]
```

In general, the user can select between these two possibilities.

# 6 Generation of Velocity and Acceleration Expressions

This section states particulars of the computation of velocity and acceleration expressions for each joint coordinate of a single loop. While the problem of establishing velocity and acceleration relationships in multibody systems poses no fundamental difficulties, the question of finding the most efficient one is still an open issue. Therefore known methods for velocity transformations were compared and the most efficient ones were chosen for subsequent implementation in the automatic processing code. Hereby, the following two basic procedures were considered:

(a) Differentiation of position equations with respect to time (*direct differentiation method*).

(b) Formulation of the closure equations at velocity level and resolution of unknowns by linear algebra methods or by geometric projections (*velocity closure approach*)

In the following, these two methodologies are discussed and compared in more detail for the case of a spatial four-bar mechanism.

## 6.1 Description and Comparison of Methods for Generation of Velocity Equations

Direct differentiation of the relative position equations with respect to time represents the most straight-forward method of obtaining velocity and acceleration expressions: as the position equations are known after applying the procedures described above, an analytic differentiation is readily carried out with built-in functions of Mathematica. Moreover, as the equations produced by the position resolution code come in cascade form, the resulting equations for velocity and acceleration exhibit a triangular structure and are thus very easy to resolve. As a consequence, one would expect that this method renders not only a simple, but also a very efficient approach to velocity and acceleration analysis. However, it turns out that, what concerns efficiency, this method ranked last in a comparison to other methods described below.

A second method for obtaining velocity expressions is the *velocity closure approach.* Here, one divides the loop in two branches, or *paths*, $\mathcal{P}_I$ and $\mathcal{P}_{II}$ and compares the velocities at the tips, or some selected components of these. Both paths have the same base frame $\mathcal{K}_l$ and the same tip frame $\mathcal{K}_k$ and contain together all transformations defined within the loop. The *twist*, i.e., the six-dimensional vector

comprising angular and linear velocity of the tip frame with respect to the base frame, can be computed in terms of the joint rates $[\dot{\underline{\beta}}]_{\mathcal{P}_I}$ and $[\dot{\underline{\beta}}]_{\mathcal{P}_{II}}$ of the branches $\mathcal{P}_I$ and branch $\mathcal{P}_{II}$, respectively. From the equality of the corresponding expressions, one obtains the system of six linear equations

$$[{}^l\boldsymbol{J}_k]_{\mathcal{P}_I}\,[\dot{\underline{\beta}}]_{\mathcal{P}_I} = [{}^l\boldsymbol{J}_k]_{\mathcal{P}_{II}}\,[\dot{\underline{\beta}}]_{\mathcal{P}_{II}} \quad . \tag{53}$$

Here, $[{}^l\boldsymbol{J}_k]_{\mathcal{P}_I}$ and $[{}^l\boldsymbol{J}_k]_{\mathcal{P}_{II}}$ represent the *Jacobians* of the branches $\mathcal{P}_I$ and $\mathcal{P}_{II}$, respectively, mapping the corresponding joint rates to the velocity twist of the tip frame.

The system of equations (53) can be readily resolved for any set of six joint rates provided that the resulting matrix of coefficients is non-singular. However, in order to minimize the number of operations, one is interested in finding a representation that implies the largest possible number of vanishing coefficients in $[{}^l\boldsymbol{J}_k]_{\mathcal{P}_I}$ and $[{}^l\boldsymbol{J}_k]_{\mathcal{P}_{II}}$. Such patterns of vanishing coefficients are determined by one or more of the following choices:

(1) the tip frame employed as end point of the branches,

(2) the base frame employed as start point of the branches,

(3) the frame in which angular and linear velocity vectors are decomposed, and

(4) the representative point at which the linear velocity of the tip frame is measured.

The effect of these parameters on the structure of the Jacobians is very difficult to foresee, so one has to proceeded largely on a heuristic basis.

Our basic heuristic rule for making an appropriate selection of velocity measurement frames is based on the structure of the loop decomposition effected during the position analysis. Hereby, taking the frame at the end of $\hat{\mathrm{A}}_A$ as the common tip of the two branches $\mathcal{P}_I$ and $\mathcal{P}_{II}$ will guarantee that at least three rows of the Jacobian vanish identically for the columns of the variables contained in $\hat{\mathrm{A}}_A$. Moreover, locating the base frame of the two branches at the start of the transformation sequence $\hat{\mathrm{A}}_B$ will also have positive effect on the filling of the Jacobian. The validity of these rules was confirmed by a number of examples.

The issue of selection of a frame of decomposition and of a representative point for linear velocity was solved by implementing two of what was found to be the most efficient methods for Jacobian calculation (Kecskeméthy 1993a, Krupp 1992): a tip-frame oriented method (Orin and Schrader 1983) and a base-frame oriented approach (Waldron 1981). The first one decomposes all vectors with respect to the tip frame, and takes also its origin as representative point for the linear velocity. In

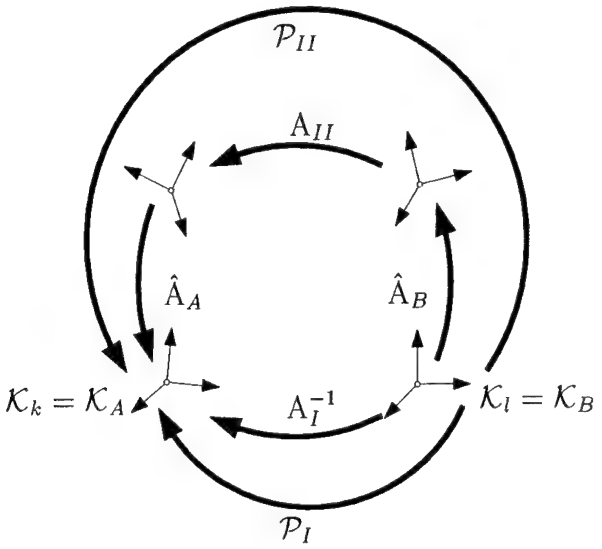



Figure 25: Decomposition of a single loop for velocity analysis

the base-frame oriented method, the velocity vectors are decomposed in the base frame, and the linear velocity of the tip is expressed in terms of the point momentarily coinciding with the origin of the base frame. The present implementation of these two methods was integrated in the general code, featuring again automatic mapping of common subexpressions to intermediate variables to minimize the number of operations.

The effect of appropriate velocity equation generation is illustrated by the following example.

### 6.1.1 Velocity Analysis of a Spatial Four-Bar Mechanism

The spatial four-bar mechanism depicted in Fig. 26 consists of two revolute joints, one universal joint and one spherical joint. Each joint implements one or more rotations about axes represented by unit vectors $\underline{u}_i$ and measured by joint coordinates $\beta_i$ , respectively. The joint coordinate $\beta_1$ is regarded as input of the loop, while the coordinates $\beta_2, \ldots, \beta_6$ are viewed as dependent ones. The loop decomposition addressed in Fig. 25 consists here in selecting the spherical joint, which comprises three rotations leaving the point of intersection of the corresponding axes invariant, as $\hat{\mathrm{A}}_A$ and the universal joint, which comprises two rotations leaving the point of intersection of the corresponding axes invariant, as $\hat{\mathrm{A}}_B$ . Thus, the remaining transformations are produced by the coupler as $\mathrm{A}_{II}$ , and the input lever, the joint with axis $\underline{u}_1$ , the base, the joint with axis $\underline{u}_2$ , and the output lever, as $\mathrm{A}_I^{-1}$ .

The two branches for velocity closure formulation are now chosen to be those starting at the fork of the universal joint and ending at the ball of the ball-and-socket

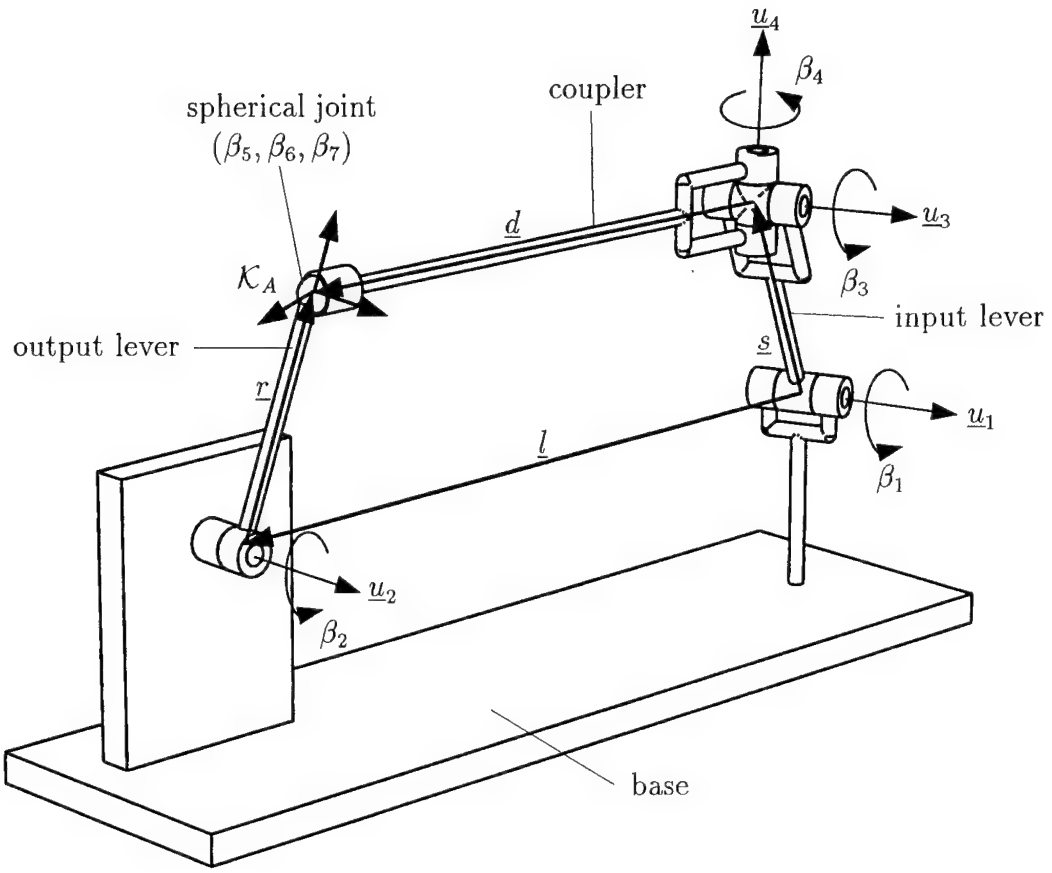



Figure 26: A spatial four-bar mechanism

joint. By taking the tip-oriented Jacobian formulation, the closure condition

$$[{}^B\boldsymbol{J}_A]_{\mathcal{P}_I}\,[\,\dot{\beta}_1\,,\,\dot{\beta}_2\,]^T = [{}^B\boldsymbol{J}_A]_{\mathcal{P}_{II}}\,[\,\dot{\beta}_3\,,\,\dot{\beta}_4\,,\,\dot{\beta}_5\,,\,\dot{\beta}_6\,,\,\dot{\beta}_7\,]^T \quad . \tag{54}$$

is obtained, where

$$\begin{aligned}
[{}^B\boldsymbol{J}_A]_{\mathcal{P}_I} &= \left[\begin{array}{cc} {}^A\underline{u}_1 & {}^A\underline{u}_2 \\ \underbrace{{}^A\underline{u}_1 \times ({}^A\underline{l} + {}^A\underline{r})}_{\dot{\beta}_1} & \underbrace{{}^A\underline{u}_2 \times {}^A\underline{r}}_{\dot{\beta}_2} \end{array}\right] , \\
[{}^B\boldsymbol{J}_A]_{\mathcal{P}_{II}} &= \left[\begin{array}{ccccc} {}^A\underline{u}_3 & {}^A\underline{u}_4 & {}^A\underline{u}_5 & {}^A\underline{u}_6 & {}^A\underline{u}_7 \\ \underbrace{{}^A\underline{u}_3 \times {}^A\underline{d}}_{\dot{\beta}_3} & \underbrace{{}^A\underline{u}_4 \times {}^A\underline{d}}_{\dot{\beta}_4} & \underbrace{\underline{0}}_{\dot{\beta}_5} & \underbrace{\underline{0}}_{\dot{\beta}_6} & \underbrace{\underline{0}}_{\dot{\beta}_7} \end{array}\right] .
\end{aligned} \tag{55}$$

Here, $\mathcal{K}_A$ denotes the frame at the ball of the spherical joint, while quantities to which the left superscript $^A$ has been added represent vectors decomposed with respect to frame $\mathcal{K}_A$. Notice that, in Eq. (55), the three lower rows of the columns corresponding to $\dot{\beta}_5$, $\dot{\beta}_6$ and $\dot{\beta}_7$ vanish. This is because the representative point for the linear velocity is invariant with respect to the corresponding rotations. Further decomposition of the involved vectors leads to the following representations of the Jacobians

$$[^B\boldsymbol{J}_A]_{\mathcal{P}_I} = \begin{bmatrix} {}^Au_{1x} & 1 \\ {}^Au_{1y} & 0 \\ {}^Au_{1z} & 0 \\ {}^A\kappa_{1x} & 0 \\ {}^A\kappa_{1y} & -r \\ {}^A\kappa_{1z} & 0 \end{bmatrix} \quad , \quad [^B\boldsymbol{J}_A]_{\mathcal{P}_{II}} = \begin{bmatrix} {}^Au_{3x} & {}^Au_{4x} & {}^Au_{5x} & 0 & 1 \\ {}^Au_{3y} & {}^Au_{4y} & {}^Au_{5y} & \cos\beta_7 & 0 \\ {}^Au_{3z} & {}^Au_{4z} & {}^Au_{5z} & \sin\beta_7 & 0 \\ {}^A\kappa_{3x} & {}^A\kappa_{4x} & 0 & 0 & 0 \\ {}^A\kappa_{3y} & {}^A\kappa_{4y} & 0 & 0 & 0 \\ {}^A\kappa_{3z} & {}^A\kappa_{4z} & 0 & 0 & 0 \end{bmatrix} . \quad (56)$$

Here, the symbols ${}^A\kappa_{i\nu}$, $i \in \{1,\ldots,6\}$ and $\nu \in \{x,y,z\}$, denote shortcuts for the components of the vector products arising in Eq. (55). Here, the conditions ${}^A\underline{u}_7 = \underline{e}_x$ and ${}^A\underline{r} = r\,\underline{e}_z$ have been worked in.

With the expressions derived above, the following three methods for determining velocity dependencies are basically possible:

1. [Direct Differentiation Method]. Direct differentiation of equations (28) with respect to time. This procedure needs no calculation of Jacobians.

2. [Projection Method]. After calculating the Jacobians using the method described in Orin and Schrader 1983 and Waldron 1981, the closure equations can be established in the form (54). Then, the following steps are effected:

   - Projection of the last three rows of the Jacobian, which represent the translational velocity, along vector $\underline{d}$. This leads to the following equation

   $$[\dot{\beta}_1{}^A\underline{u}_1 \times ({}^A\underline{l} + {}^A\underline{r}) + \dot{\beta}_2{}^A\underline{u}_2 \times {}^A\underline{r}] \cdot ({}^A\underline{l} + {}^A\underline{r} - {}^A\underline{s}) =$$
   $$\underbrace{[\dot{\beta}_3{}^A\underline{u}_3 \times {}^A\underline{d} + \dot{\beta}_4{}^A\underline{u}_4 \times {}^A\underline{d}] \cdot {}^A\underline{d}}_{\underline{0}} \quad , \quad (57)$$

   where the unknown joint velocities $\dot{\beta}_3$ and $\dot{\beta}_4$ are eliminated. The resulting scalar equation can be solved for $\dot{\beta}_2$.

   - Simultanously resolution of two of the last three equations for $\dot{\beta}_3$ and $\dot{\beta}_4$.

   - Simultanously resolution of the second and the third equation for $\dot{\beta}_5$ and $\dot{\beta}_6$.

- Resolution of the first equation for $\dot{\beta}_7$.

3. [Coefficient Analysis Method]. By taking into consideration all vanishing entries in Eq. (56), one can first solve rows 4 and 6 for $\dot{\beta}_3$ and $\dot{\beta}_4$ ; then row 5 for $\dot{\beta}_2$ ; then rows 2 and 3 for $\dot{\beta}_5$ and $\dot{\beta}_6$ ; and finally row 1 for $\dot{\beta}_7$ . Note that here only $2 \times 2$ subsystems of linear equations need to be solved, and that there is no need for projection operations. Also, note that the sequence of resolution is reversed for the first three unknowns compared to the previous method. However, it should be emphasized that this special resolution scheme resulting from the special geometry of the loop can not be generalized for arbitrary loop geometries.

The comparison of the three methods yielded the total operation count shown in Table 3, with all elementary operations (addition, subtraction, multiplication and division) counted equally. Trigonometric operations were not taken into account, as they are already computed in the position analysis.

| Method | Number of Operations | Improvement |
|---|---|---|
| 1 | 297 | reference |
| 2 | 183 | 38.4 |
| 3 | 139 | 53.2 |

Table 3: Comparison of computational efficiency for different velocity analysis methods

Clearly, the *coefficient analysis method* yields the best results for this example, although this method is the most difficult to automatize. However, the table shows that the *projection method* is only slightly less efficient than the previous method, making it a good candidate for automatic velocity analysis. While this result can not be generalised to all single loops it was decided to implement both methods and then choose the best on a loop-by-loop basis. This will be explained in detail in section 6.2. Note, finally, that the "straight-forward" method of direct differentiation yields the worst results, though it is very easy to implement.

## 6.2 Implementation of Velocity Processing Procedures

The previous example shows that the direct consideration of the pattern of zeroes in the Jacobian and the projection method yield the most efficient results. Therefore, both methods were implemented in the current version of the symbolic kinematics processing program. Below we reproduce some heuristic rules for selecting the most appropriate method based on a more detailed analysis of the structure of Jacobians.

### 6.2.1 General Structure of Jacobians

As was mentioned earlier, there are several possibilities of evaluating the columns of the Jacobian, two of the most efficient ones being the methods of Waldron 1981 (base frame oriented) and Orin and Schrader 1983 (tip frame oriented). These methods will be described below for a serial chain of $n+1$ bodies and $n$ revolute or prismatic joints attached to an inertial base.

Let there be a reference system $\mathcal{K}_i, i = 1,\ldots,n$ attached to each joint $\mathcal{G}_i$ *before* the transformation, where the joint axis $\underline{u}_i$ is collinear with one of the axes of $\mathcal{K}_i$. An additional reference system $\mathcal{K}_{n+1}$ is attached at the tip of the chain *after* the corresponding joint transformation. The Jacobian can be decomposed either with respect to the tip frame $\mathcal{K}_{n+1}$ or with respect the base frame $\mathcal{K}_1$.

In the first case, the Jacobian results as

$$ {}^1\boldsymbol{J}_{n+1}^{(n+1)} = {}^1\boldsymbol{J}_{n+1} = \begin{bmatrix} \bar{\sigma}_1\, {}^{n+1}\underline{u}_1 & \cdots & \bar{\sigma}_n\, {}^{n+1}\underline{u}_n \\ \sigma_1\, {}^{n+1}\underline{u}_1 + \bar{\sigma}_1\, {}^{n+1}\underline{\chi}_1 & \cdots & \sigma_n\, {}^{n+1}\underline{u}_n + \bar{\sigma}_n\, {}^{n+1}\underline{\chi}_n \end{bmatrix} , \tag{58} $$

where ${}^{n+1}\underline{\chi}_i = {}^{n+1}\underline{u}_i \times {}^{n+1}_i\underline{r}_{n+1}$. The involved vectors are computed by the following recursive algorithm (Orin and Schrader 1983):

$$ \left.\begin{array}{lcll} {}^{n+1}R_{n+1} & = & I & \\ {}^{n+1}R_{i-1} & = & {}^{n}R_i\, {}^{i-1}R_i^T \ , & i = n+1, n, \ldots, 2 \\ {}^{n+1}\underline{u}_i & = & {}^{n+1}R_i\, {}^{i}\underline{u}_i \ , & i = 1, 2, \ldots, n \\ {}^{n+1}_{n+1}\underline{r}_{n+1} & = & \underline{0} & \\ {}^{n+1}_{i-1}\underline{r}_{n+1} & = & {}^{n+1}_{i}\underline{r}_{n+1} + {}^{n+1}R_{i-1}\, {}^{i-1}_{i-1}\underline{r}_i \ , & i = n+1, n, \ldots, 2 \\ {}^{n+1}\underline{\chi}_i & = & {}^{n+1}\underline{u}_i \times {}^{n+1}_i\underline{r}_{n+1} \ , & i = 1, 2, \ldots, n \end{array}\right\} . \tag{59} $$

In the second case, the Jacobian is obtained as

$$ {}^1\boldsymbol{J}_{n+1}^{(1)} = \begin{bmatrix} \bar{\sigma}_1\, {}^{1}\underline{u}_1 & \cdots & \bar{\sigma}_n\, {}^{1}\underline{u}_n \\ \sigma_1\, {}^{1}\underline{u}_1 + \bar{\sigma}_1\, {}^{1}\underline{\chi}_1 & \cdots & \sigma_n\, {}^{1}\underline{u}_n + \bar{\sigma}_n\, {}^{1}\underline{\chi}_n \end{bmatrix} , \tag{60} $$

with ${}^1\underline{\chi}_i = {}^1\underline{u}_i \times {}^1_i\underline{r}_1$. Here the involved vectors are computed using the following

recursive algorithm (Waldron 1981):

$$\left.\begin{array}{lcll} {}^1R_1 & = & I & \\ {}^1R_i & = & {}^1R_{i-1}\,{}^{i-1}R_i \;, & i=2,3,\ldots,n \\ {}^1\underline{u}_i & = & {}^1R_i\,{}^i\underline{u}_i \;, & i=1,2,\ldots,n \\ {}^1_1\underline{r}_1 & = & \underline{0} & \\ {}^1_1\underline{r}_i & = & {}^1_1\underline{r}_i + {}^1R_i\,{}^i_1\underline{r}_i \;, & i=1,2,\ldots,n \\ {}^1\underline{\chi}_i & = & -{}^1\underline{u}_i \times {}^1_1\underline{r}_i \;, & i=1,2,\ldots,n \end{array}\right\} \quad . \tag{61}$$

### 6.2.2 Basic Cases Leading to Simplification of Velocity Expressions

Optimized expressions for the dependent joint velocities and accelerations are of particular advantage when the subchains $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$ described above comprise together five unknowns. These cases occur when the geometric elements left invariant by $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$ are either a *point* or a *plane*. There are four possible combinations for relating these two elements to the chains $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$. For all four cases, if the tip frame of $\hat{\mathrm{A}}_A$ is used for decomposition of the resulting vectors, and if its origin is used for describing its translational velocity, there will always exist a $3 \times 3$ submatrix of the Jacobian that vanishes. The only difference between these four cases lies in the ensuing pattern of vanishing coefficients, which is discussed next.

**$\hat{\mathbf{A}}_A$ and $\hat{\mathbf{A}}_B$ both leave a *point* invariant**
In this case, the transformation sequences $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$ correspond to a spherical joint and a universal joint, respectively. This case was illustrated by the example of a spatial four-bar mechanism. If the Jacobian contains enough zero elements to allow a partitioned resolution with blocks not larger than $2 \times 2$, the decomposition of the *coefficient analysis method* is employed, otherwise the *projection method* is used.

**Invariant element of $\hat{\mathbf{A}}_A$ is a *point*; invariant element of $\hat{\mathbf{A}}_B$ is a *plane***
Here, the transformation $\hat{\mathrm{A}}_A$ represents again a spherical joint, while the transformation $\hat{\mathrm{A}}_B$ corresponds to a planar joint. The velocity of the origin of the tip frame of $\hat{\mathrm{A}}_A$ does again not depend on the three joint rates of the spherical joint, so these are eliminated from the three translational velocity equations. The systems can be handled basically in the same way as described in section 6.2.2. However, instead of projecting the translational velocity along the distance vector between $\hat{\mathrm{A}}_A$ and $\hat{\mathrm{A}}_B$, one carries out this projection now along the normal to the plane invariant to the transformation $\hat{\mathrm{A}}_B$.

**Invariant element of $\hat{\mathrm{A}}_A$ is a *plane*; invariant element of $\hat{\mathrm{A}}_B$ is a *point***
For this case, only the *coefficient analysis method* is efficient. This is illustrated by the example of spatial mechanism with a planar joint shown in Fig. 27. The mechanism consists of two rotational joints, one universal joint and one planar joint. The planar joint is replaced by a combination of three parallel rotational joints with joint axis $\underline{u}_5$, and joint variables $\beta_5$, $\beta_6$ and $\beta_7$. Of the seven joint variables, the angle $\beta_1$ is regarded as input of the loop, while the variables $\beta_2, \ldots, \beta_7$ are viewed as dependent. The joints are denoted below by the index of the corresponding joint variables.

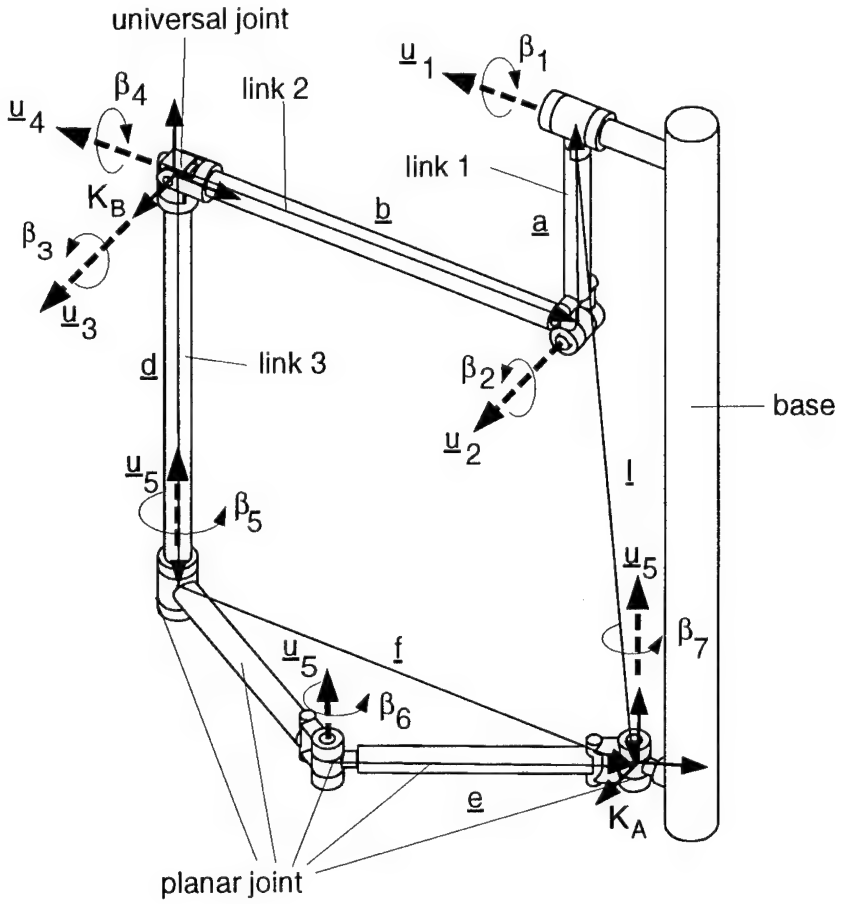



Figure 27: A spatial mechanism with a planar joint

The loop dissection introduced above amounts to selecting the planar joint, whose three variable rotations leave the plane perpendicular to the axis $\underline{u}_5$ invariant, as subchain $\hat{\mathrm{A}}_A$, and the universal joint, whose two variable rotations leave the point of intersection of axes $\underline{u}_3$ and $\underline{u}_4$ invariant, as subchain $\hat{\mathrm{A}}_B$. Then, the link 3 acts as the subchain $\mathrm{A}_{II}$, and the rest of the transformations, i.e., link 2, the joint with axis $\underline{u}_1$, link 1, and the base link, embody together the subchain $\mathrm{A}_I^{-1}$.

The two branches for the velocity closure condition comprise now the two chains starting at the reference system $\mathcal{K}_B$ at the left end of link 2, with one of its axes aligned with the $\underline{u}_3$-axis of the universal joint, to the frame $\mathcal{K}_A$ centered at the origin of joint 7 and connected to the base link. One of these, $\mathcal{P}_{II}$, traverses the chain comprising the universal joint, the link 3 and the planar joint, while the other one, $\mathcal{P}_I$, comprises links 2 and 1, joints 2 and 1, and the base link. The closure equation for velocities thus reads

$$[{}^B\boldsymbol{J}_A]_{\mathcal{P}_I}\,[\dot{\beta}_1\,,\,\dot{\beta}_2]^T = [{}^B\boldsymbol{J}_A]_{\mathcal{P}_{II}}\,[\dot{\beta}_3\,,\,\dot{\beta}_4\,,\,\dot{\beta}_5\,,\,\dot{\beta}_6\,,\,\dot{\beta}_7]^T \quad , \tag{62}$$

with

$$\begin{aligned}
[{}^B\boldsymbol{J}_A]_{\mathcal{P}_I} &= \left[\underbrace{{}^{A}\underline{u}_1 \times {}^{A}\underline{l}}_{\dot{\beta}_1} \quad \underbrace{{}^{A}\underline{u}_2 \times ({}^{A}\underline{a} + {}^{A}\underline{l})}_{\dot{\beta}_2}\right] \quad , \\
[{}^B\boldsymbol{J}_A]_{\mathcal{P}_{II}} &= \left[\underbrace{{}^{A}\underline{u}_3 \times ({}^{A}\underline{d} + {}^{A}\underline{f})}_{\dot{\beta}_3} \quad \underbrace{{}^{A}\underline{u}_4 \times ({}^{A}\underline{d} + {}^{A}\underline{f})}_{\dot{\beta}_4} \quad \underbrace{{}^{A}\underline{u}_5 \times {}^{A}\underline{f}}_{\dot{\beta}_5} \quad \underbrace{{}^{A}\underline{u}_5 \times {}^{A}\underline{e}}_{\dot{\beta}_6} \quad \underbrace{{}^{A}\underline{u}_5 \; \underline{0}}_{\dot{\beta}_7}\right] \quad .
\end{aligned} \tag{63}$$

All vectors are decomposed in frame $\mathcal{K}_A$, whose $z$-axis is aligned with the normal to the plane of the planar joint, $\underline{u}_5$. This choice is arbitrary, but it can nevertheless be assumed that the normal to the plane will be parallel to one of the coordinate axes of $\mathcal{K}_A$, because otherwise the transformation at the joint 7 would not be elementary. Now, the angular velocity vectors of joints 5, 6, and 7 all have a nonzero entry only in the $z$-component, which is equal to the joint rate, so the corresponding block of the two upper rows of the Jacobian vanish. Similarly, one can appreciate that joints 5, 6, and 7 do not produce a velocity component in $z$-direction, yielding three zero entries in the last row of the Jacobian. Thus, again a $3 \times 3$ submatrix of the Jacobian has vanishing coefficients. Furthermore, as the reference point for $\mathcal{K}_A$ lies on the axis of joint 7, the last three entries of the last column of the Jacobian also vanish. Finally, as the axis of joint 4 always remains perpendicular to the $z$-axis of $\mathcal{K}_A$, the $z$-component of the translational velocity induced by joint 4 is also zero. Thus, the Jacobians for the present example takes the form

$$[{}^B\boldsymbol{J}_A]_{\mathcal{P}_I} = \begin{bmatrix} {}^{A}u_{1x} & {}^{A}u_{2x} \\ {}^{A}u_{1y} & {}^{A}u_{2y} \\ {}^{A}u_{1z} & {}^{A}u_{2z} \\ {}^{A}\kappa_{1x} & {}^{A}\kappa_{1x} \\ {}^{A}\kappa_{1y} & {}^{A}\kappa_{1y} \\ {}^{A}\kappa_{1z} & {}^{A}\kappa_{1z} \end{bmatrix} \quad , \quad [{}^B\boldsymbol{J}_A]_{\mathcal{P}_{II}} = \begin{bmatrix} {}^{A}u_{3x} & {}^{A}u_{4x} & 0 & 0 & 0 \\ {}^{A}u_{3y} & {}^{A}u_{4y} & 0 & 0 & 0 \\ {}^{A}u_{3z} & {}^{A}u_{4z} & 1 & 1 & 1 \\ {}^{A}\kappa_{3x} & {}^{A}\kappa_{4x} & {}^{A}\kappa_{5x} & {}^{A}\kappa_{6x} & 0 \\ {}^{A}\kappa_{3y} & {}^{A}\kappa_{4y} & {}^{A}\kappa_{5y} & {}^{A}\kappa_{6y} & 0 \\ {}^{A}\kappa_{3z} & 0 & 0 & 0 & 0 \end{bmatrix} \quad . \tag{64}$$

With the given pattern of zero coefficients of the Jacobians, the best resolution scheme is to solve rows 1, 2 and 6 for the unknowns $\dot{\beta}_2$, $\dot{\beta}_3$, $\dot{\beta}_4$, then rows 4 and 5 for $\dot{\beta}_5$ and $\dot{\beta}_6$ and finally row 3 for $\dot{\beta}_7$.

**Both $\hat{\mathbf{A}}_A$ and $\hat{\mathbf{A}}_B$ leave a *plane* invariant**
This case corresponds previous one, and can be treated correspondingly.

## 6.3 Generation of Acceleration Expressions

From the velocity closure conditions, one can readily derive the acceleration closure conditions by direct time-differentiation as

$$[{}^l\boldsymbol{J}_k]_{\mathcal{P}_I}\,[\underline{\ddot{\beta}}]_{\mathcal{P}_I} + [{}^l\dot{\boldsymbol{J}}_k]_{\mathcal{P}_I}\,[\underline{\dot{\beta}}]_{\mathcal{P}_I} = [{}^l\boldsymbol{J}_k]_{\mathcal{P}_{II}}\,[\underline{\ddot{\beta}}]_{\mathcal{P}_{II}} + [{}^l\dot{\boldsymbol{J}}_k]_{\mathcal{P}_{II}}\,[\underline{\dot{\beta}}]_{\mathcal{P}_{II}} \quad . \tag{65}$$

Note that $[{}^l\dot{\boldsymbol{J}}_k]_{\mathcal{P}_I}\,[\underline{\dot{\beta}}]_{\mathcal{P}_I}$ and $[{}^l\dot{\boldsymbol{J}}_k]_{\mathcal{P}_{II}}\,[\underline{\dot{\beta}}]_{\mathcal{P}_{II}}$ can be calculated from vector products as described in equations (18) and (19). Thus, no analytical differentiations are necessary at this point.

After determining $[{}^l\dot{\boldsymbol{J}}_k]_{\mathcal{P}_I}\,[\underline{\dot{\beta}}]_{\mathcal{P}_I}$ and $[{}^l\dot{\boldsymbol{J}}_k]_{\mathcal{P}_{II}}\,[\underline{\dot{\beta}}]_{\mathcal{P}_{II}}$, the unknown accelerations can be determined using the same resolution scheme as for the velocities. Thus one can restrict attention to the treatment of the velocity equations (53), and apply the resulting techniques subsequently to the acceleration problem.

## 6.4 Example: Wheel Suspension of a Trailer

For the trailer wheel suspension the following sequence of equations is generated for the unknown joint velocities

```
ADH14[2, 4] -> v3y Cos[gamma] - v3z Sin[gamma]
ADH14[3, 4] -> -(v3z Cos[gamma]) - v3y Sin[gamma]
                 s'
alpha' -> -----------
           ADH14[2, 4]
          ADH14[3, 4] alpha'
beta' -> ------------------
                    s
gamma' -> -alpha' + beta'
```

Similarly for the accelerations one obtains

```
JACR6[6] -> -(s beta')
JACR8[5] -> -(ADH14[2, 4] alpha')
JACR8[6] -> -(ADH14[3, 4] alpha')
JACR9[5] -> -(ADH14[2, 4] alpha') - ADH14[2, 4] gamma'
JACR9[6] -> -(ADH14[3, 4] alpha') - ADH14[3, 4] gamma'
JACR10[5] -> -(JACR9[5] alpha') - JACR8[5] gamma'
JACR10[6] -> -(JACR9[6] alpha') - JACR8[6] gamma'
```

```
JACR7[6] -> -(JACR6[6] beta')
           -JACR10[6] + JACR7[6] + s''
alpha'' -> --------------------------
                    ADH14[2, 4]
          -JACR10[5] + ADH14[3, 4] alpha''
beta'' -> --------------------------------
                         s
gamma'' -> -alpha'' + beta''
```

The shortcuts JACR$k$[$i$] represent intermediate terms arising in the evaluation of the velocity and acceleration expressions, and can be interpreted to be the $i$th element of a column, or its first time-derivative, of a Jacobian. Note that the velocity and acceleration expressions make use of the intermediate expressions generated in the previous steps.

# 7 Generation and Solution of Kinematical Networks

After decomposing a general multiloop system in its constituents, i. e., in a set of independent loops, and generating the local kinematical equations for the individual loops, the next step is to re-assemble the local kinematical equations into a global set of equations. A method of accomplishing this is the method of kinematical transformers introduced by the author in Kecskeméthy 1993b. The basic idea of this method is to regard each loop as a nonlinear transmission element that maps the values of the independent joint variables $\underline{q}$ to the values of the corresponding dependent variables $\underline{\beta}^{exi}$. In this way, the local geometry of the loop can be neglected when it is being assembled into the global system and all that needs to be regarded are the number of independent variables, termed the local degree of freedom $f_L$ of the loop, and the number of dependent variables, termed its outputs. Note that in the general case the number of dependent variables is six, while for the case of planar or spherical mechanisms the number of outputs is three. The so constructed element is denominated a *kinematical transformer*. With the model of the kinematical transformer, the task of generating the global kinematical equations for a multiloop system consists in recognizing in which way the inputs and outputs of the individual kinematical transformers are concatenated. As it will be seen, this concatenation is achieved by linear equations. Thus the assembly of the individual kinematical transformers into the global system can be reproduced by a network of linearly coupled nonlinear transmission elements, which we term a kinematical network. From this network, it is rather simple to gather further information about the system, as for example the solution flow, i. e., the propagation of the motion of the global inputs through the mechanism, or, if applicable, the recursive solution flow, which is the order in which the loops can be processed such as to obtain a recursive solution. Below we reproduce a short overview of the method for easier reference. Results of the corresponding Mathematica implementation shall be presented in Section **11**.

## 7.1 Determination of Loop-Coupling Conditions

The joint variables within a loop are denoted by $\beta_1, \ldots, \beta_{n_\beta(L)}$ when it is not clear which of the coordinates to use as inputs and which to use as outputs. It is clear that by modeling each loop independently of the others a redundant set of joint coordinates is introduced and that a dependency between the joint coordinates defined in the different loops will result. This dependency can be formulated quite simply if one regards only elementary joints (i. e. prismatic (P), revolute (R) or screw (H) joints) as joints where a coupling can occur. This pre-assumption is not very severe because, as it is known, all technical joints can be decomposed into a

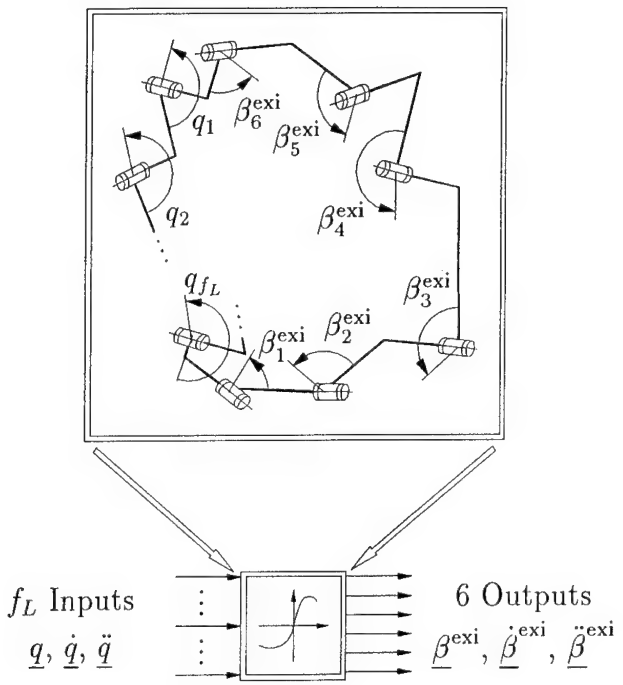



Figure 28: Model of the general kinematical transformer

sequence of elementary joints. However, it will be allowed that multiple joints $\mathcal{G}$ connecting $n_B(\mathcal{G}) \geq 2$ bodies $\mathcal{B}_i$ occur (Fig. 29).

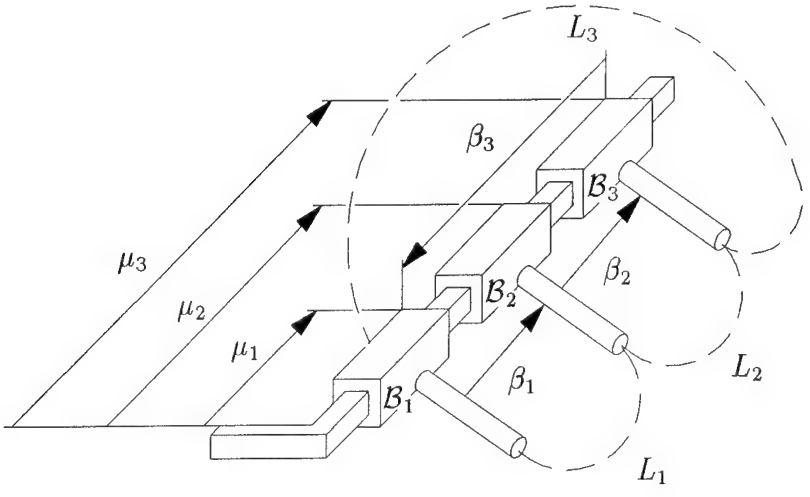



Figure 29: A joint connecting several bodies

Now, assume that the joint is part of $n_L(\mathcal{G})$ independent loops $L_i$. Let $\mu_i$ denote

the position of body $\mathcal{B}_i$ in the natural coordinate system of the joint with respect to a given reference point, and $\beta_k$ a relative joint coordinate defined in loop $L_k$ describing the relative position of bodies $\mathcal{B}_{i(k)}$ and $\mathcal{B}_{j(k)}$. Each loop connected to the joint introduces a linear equation defining a relative joint coordinate as the difference of the relative position of two of the bodies connected by the joint:

$$\mu_{i(k)} - \mu_{j(k)} = \beta_k + \alpha_k \quad , \; k = 1, \ldots, n_L(\mathcal{G}) \quad . \tag{66}$$

After collecting all of these equations, and supplementing them with a equation for the definition of the reference point within the joint, e. g.

$$\mu_1 = 0 \quad , \tag{67}$$

one obtains a system of $n_L(\mathcal{G}) + 1$ linear equations

$$P\,\underline{\mu} = \underline{\hat{\beta}} + \underline{\hat{\alpha}} \tag{68}$$

with

$$\underline{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_{n_{\mathcal{B}(\mathcal{G})}} \end{bmatrix} \quad , \quad \underline{\hat{\beta}} = \begin{bmatrix} 0 \\ \beta_1 \\ \vdots \\ \beta_{n_L(Joint)} \end{bmatrix} \quad , \quad \underline{\hat{\alpha}} = \begin{bmatrix} 0 \\ \alpha_1 \\ \vdots \\ \alpha_{n_L(Joint)} \end{bmatrix} \tag{69}$$

defining $\underline{\mu}$ as linear function of $\underline{\hat{\beta}}$. Note that $P$ is a $(n_L(\mathcal{G}) + 1) \times n_{\mathcal{B}(\mathcal{G})}$ matrix, where, in general $(n_L(\mathcal{G})+1) \neq n_{\mathcal{B}(\mathcal{G})}$. Thus, in order for Eq. (68) to have a solution, it must be ensured that the vector $\underline{\hat{\beta}}+\underline{\hat{\alpha}}$ lies in the range of matrix $P$. This condition can be expressed with the help of the orthogonal complement $B$ of $P$ (Halmos 1987), defined as

$$B^{\mathrm{T}} P = 0 \quad . \tag{70}$$

It follows

$$B^{\mathrm{T}} \left( \underline{\hat{\beta}} + \underline{\hat{\alpha}} \right) = 0 \quad . \tag{71}$$

The number of independent columns of matrix $B$ and thus the number of linear equations which the joint coordinates $\underline{\hat{\beta}}$ must fulfil is equal to $n_L(Joint) + 1 - r$, where $r$ is the rank of $P$. This rank is equal to $n_{\mathcal{B}(\mathcal{G})}$ minus the increase of the number of connected components resulting from the original system when the joint is removed. Thus, for a 2-connected graph, i. e. one in which at least two joints have

to be removed in order for the mechanism to fall into two parts (which is the normal case in mechanism analysis), the rank of $P$ is $n_{\mathcal{B}(\mathcal{G})}$ and the number of coupling conditions at a joint can be determined by the formula:

$$p(\mathcal{G}_i) = n_L(\mathcal{G}_i) - n_{\mathcal{B}(\mathcal{G}_i)} + 1 \quad . \tag{72}$$

It is interesting to note that matrix $P$, after removing the first line, can be interpreted as the incidence matrix of a graph whose nodes represent the coordinates $\mu_i$ and the edges the variables $\beta_k$ directed from $\mu_{j(k)}$ to $\mu_{i(k)}$. Then, matrix $B^{\mathrm{T}}$ defined by Eq. (70) corresponds just to the cycle matrix of this graph. Thus one can obtain the coupling conditions using the same algorithms as described in Section **4**. Note also that from this property it follows directly that the elements of matrix $B$ are only $+1$, $-1$, $0$, so that the coupling conditions are just signed sums of joint variables.

It can be shown that the counting rule (72) yields enough coupling conditions between the independently modeled loops so as to allow to assemble them correctly to general systems (Kecskeméthy 1993a). Indeed, the following identity holds:

$$f = \sum_{j=1}^{n_L} f_{L_j} - \sum_{i=1}^{n_{\widehat{G}}} p(\mathcal{G}_i) \quad , \tag{73}$$

where $f$ is the global degree of freedom of the mechanism, $f_{L_j}$ is the local degree of freedom of loop $L_j$, and $n_{\widehat{G}}$ denotes the number of joints, including multiple joints. Thus, dissecting a general multibody system into individual loops and then formulating the coupling conditions at the joints yields a system of equations which is equivalent to the traditional methods of multibody kinematic analysis. Eq. (72) plays a role similar to the Grübler-Kutzbach formula of spatial kinematics.

The conditions of linear coupling derived for elementary joints can be extended to other joints as well, but these joints must fulfil two conditions: (1) it must be ensured that the result of two joint transformations applied in sequence is again a transformation which can be described by the same joint, and (2) the composition of two transformations must be commutative. Property (1) implies that the transformations associated with the joint must be subgroups of rigid-body motion, while property (2) states that these subgroups must be Abelian. Table 4 shows joints of different degrees of freedom $f_{\mathcal{G}}$ possessing these properties. Besides the elementary joints, these are the cylindrical (C), planar translational (2P) and spatial translational (3P) joints.

## 7.2 Selection of an Appropriate Solution Flow

After having established the independent loops and their couplings, for the remaining problem of determining an appropriate ordering of the global equations one can

| $f_G$ | 1 | 2 | 3 | 4–6 |
|---|---|---|---|---|
| type | R, P, H | C, 2 P | 3 P | none |

Table 4: Joints yielding linear coupling conditions

imagine the individual loops to be kinematical transformers which are connected by summing junctions representing the linear coupling equations. The task is now to find orientations for the edges connecting the loops such that a block diagram results having the following properties (Hiller and Kecskemethy 1989):

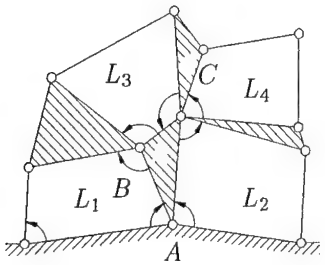
1. The number of external inputs is equal to the number of degrees of freedom of the system.
2. The number of inputs for each multibody loop $L_i$ is equal to the local degree of freedom $f_{L_i}$ of the loop.
3. Each summing junction has exactly one output.
4. There are no closed circuits.
5. The local kinematics of the transformers are recursively solvable.

It is obvious that the complete system of equations is then recursively solvable.

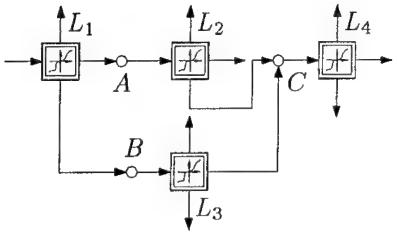
Surprisingly, for many technical applications conditions (1) through (5) can indeed be fulfilled. These systems are termed *recursively solvable systems*. Systems for which not all conditions can be fulfilled are called *non-recursively solvable systems*. The most common reason for the appearance of a non-recursively solvable system is that conditions (1) through (4) can not be accomplished. The cases for which condition (5) is violated (e. g. for the general case of a 7R-mechanism) do not occur so often in practice and shall not be regarded here.

A very simple method for finding the appropriate orientation of the edges for the case of *recursively solvable* systems is to start at the *sinks* of the system, i. e. where all edges connected to an element can be oriented *into* the element. Then, after finding such an element and orienting the edges, one removes both the element and the oriented edges and looks for the next sink, and so on. Clearly, the number of allowed input edges for a kinematical transformer is equal to its internal degree of freedom, while the number of allowed input edges for a summing junction equals the number of connections minus one. Also, there will exist branching nodes which have exactly one input. It may happen that after carrying out this algorithm some summing junctions or branching nodes remain which do not have enough inputs: this problem is fixed easily by reversing the direction of an appropriate number of edges pointing out of the element, re-orienting edges only once in order to avoid deadlocks. The resulting oriented block-diagram is denominated the "solution flow".

As an example of a recursively solvable system, a planar mechanism consisting of four interconnected planar four-bar loops is considered (Fig. 30). The redundant set of relative coordinates includes for each loop four variables. Three of these can be solved as functions of the fourth in closed form, yielding corresponding kinematical transformers. There are three linear assembly equations occuring in the joints $A$, $B$, $C$. A sequence of elements for which unoriented edges can be oriented as described above is: $L_4$, $C$, $L_3$, $L_2$, $B$, $A$, $L_1$. This sequence yields a "solution flow" which obviously is recursive. Thus the constraint equations of this system are solvable in *closed form.*

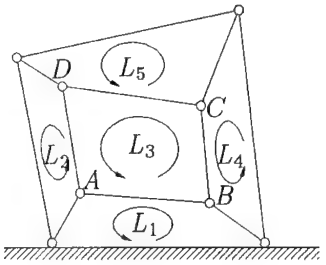



a) multibody system

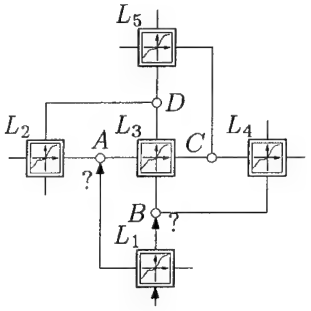



b) kinematical network

Figure 30: A recursively solvable system




a) multibody system




b) kinematical network

Figure 31: A non-recursively solvable system

An example of a non-recursively solvable system is shown in Fig. 31. The planar mechanism consists of five independent multibody loops which are again four-bar mechanisms. There are four linear assembly equations at the joints $A$, $B$, $C$, $D$. From the corresponding block diagram it is clear that the algorithm described above can not start, because there is no element which has an allowable number of inputs greater than the number of connections. This situation changes when the summing junction $D$ is removed and an additional input $\tilde{q}$ is provided. In this case the system is recursively solvable. The original system results after re-inserting the summing junction $D$, yielding an implicit equation for the determination of the function $\tilde{q}(q)$.

## 7.3 Example: The Heavy-Load Manipulator

The heavy-load manipulator was decomposed into three subsystems, each of which represented a joint unit. For each joint unit, a decoupled kinematical network is

produced. One of these kinematical networks consists of just one loop ($L_1$), while the other two contain two loops, which are coupled through a revolute joint. The resulting block diagrams are shown in Fig. 32, where $L_1$ represents the first joint unit, and $L_2$, $L_3$ and $L_4$, $L_5$ represent the second and third joint unit, respectively.

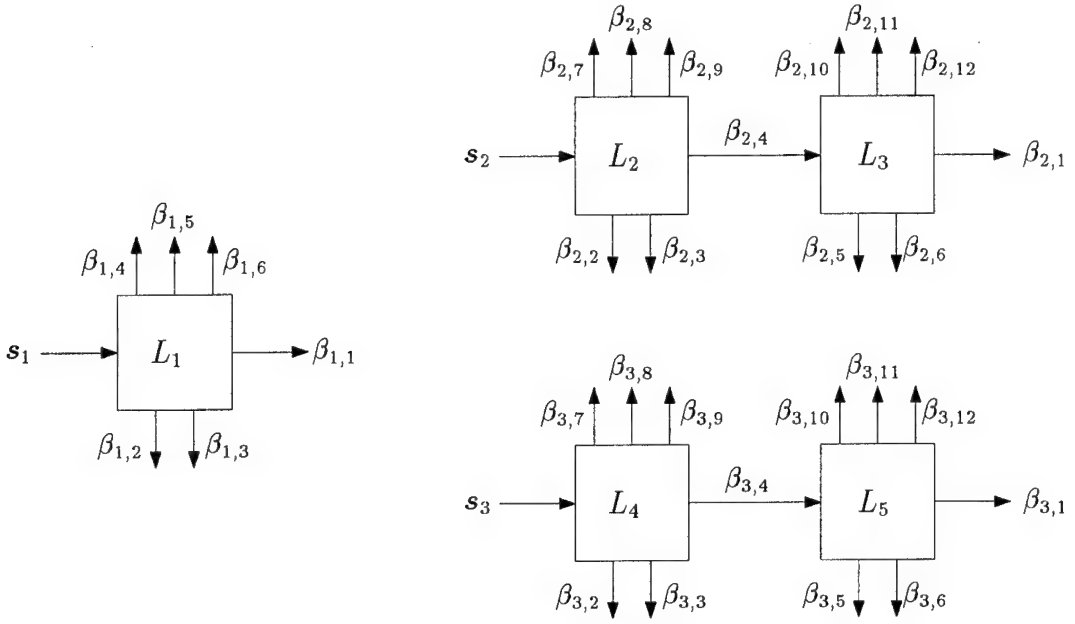



Figure 32: Kinematic networks for the heavy-load manipulator

As depicted in Fig. 32, a recursive solution flow is possible for all kinematic networks. Thus the system has overall recursively solvable relative kinematics. The solution starts with the global input coordinates which are entries for the loops $L_1$, $L_2$ and $L_4$. After solving the constrained equations for these three loops, $L_3$ and $L_5$ can be calculated using the outputs of $L_2$ and $L_4$.

# 8 Implicit Solutions for Non Recursively Solvable Subsystems

In the preceding sections, multiloop systems for which recursive solutions exist were regarded, and a corresponding methodology for the automated generation of such solutions was developed. In this section, the case in which such closed-form solutions do not exist, either because there is no recursive solution flow for the corresponding kinematical network or because at least one of the kinematical loops is not explicitly solvable, will be treated. In this case, only an iterative solution is feasible, and the topological processing is limited to finding an optimal set of closure conditions that minimize the corresponding numerical effort.

The basic idea for the determination of suitable closure conditions consists in regarding the multibody system as a tree-type mechanism to which additional constraints have been added. These additional constraints are termed *secondary joints*, while the joints of the underlying tree-type system are denoted by *primary joints*. As with the topological problems discussed in the previous sections, graph theory offers also here systematic ways of finding an appropriate topological processing scheme. Here, the problem consists of finding for a general connected graph a *spanning tree* that comprises all nodes of the original graph, but contains no cycles. The edges of the graph that are not part of the spanning tree form what is denoted the *co-tree*. Each edge of the co-tree forms together with the edges of the spanning tree exactly one loop. Thus the problem of determining a spanning tree is related to the problem of determining a fundamental cycle set. Of particular interest for the iterative solution is the notion of the *minimum weight spanning tree* (Carré 1979, Gondran and Minoux 1984). The minimum weight spanning tree is the spanning tree of a weighted graph, i. e. one in which each edge is attributed a positive *weight*, that renders the smallest value for the sum of weights of its edges. If one employs as weights the number of elementary transformations involved in each edge, the minimal weight spanning tree will produce an interconnection structure in which each body can be reached using the minimal number of transformations, and thus of operations. This will have two effects: on the one hand, each function evaluation for the iterative solution will involve a minimal number of operations and thus be more efficient; on the other hand, the number of transformations left over in the secondary joints will be maximal, thus reducing the number of constraint equations which correspond to blocked degrees of freedom of the secondary joints. As a consequence, the minimum-weight spanning tree approach can reduce dramatically the computational effort of the iterative solution.

In the following, it will be assumed that the inputs of the system have been assigned to appropriate joint variables. For this purpose, the automatic processing code determines the number of inputs that can be supplied for each cluster, and then chooses within each cluster as inputs the joints that are shared by the largest

number of loops. The latter rule has produced very good results for the processed examples.

## 8.1 Identification of Secondary Joints

Let the kinematics of the system be described again by a directed weighted graph $G = \langle K, E \rangle$, where the weight of each edge $e_{ij}$ corresponds to the number of dependent joint coordinates contained therein. Starting at any arbitrary node $\mathcal{K}_i$, a minimum weight spanning tree is obtained by constructing a sequence of tree structured graphs, $T^{(k)} = \langle K^{(k)}, F^{(k)} \rangle$ , $k = 0, 1, \ldots, n_k - 1$, defined as follows.

**1:** Set $K^{(0)} = \{\mathcal{K}_i\}$ and $F^{(0)} = \emptyset$.

**2:** For $k = 1, \ldots, n-1$, let $X^{(k)}$ be the set of edges that have exactly one endpoint in $K^{(k-1)}$, let $e^{(k)}$ be the shortest edge of $X^{(k)}$ and let $\mathcal{K}_l$ be the endpoint of $e^{(k)}$ which does not belong to $K^{(k-1)}$. Then set

$$K^{(k)} = K^{(k-1)} \cup \{\mathcal{K}_l\} \ , \tag{74}$$

$$F^{(k)} = F^{(k-1)} \cup \{e^{(k)}\} \ . \tag{75}$$

On termination, the graph $T^{(n_k-1)} = \langle K^{(n_k-1)}, F^{(n_k-1)} \rangle$ is the minimum weight spanning tree of the graph $G$ sought for.

The co-tree is obtained as the complement of $F^{(n_k-1)}$ in $E$, i. e.,

$$C = E - F^{(n_k-1)} \ . \tag{76}$$

Note that the co-tree is just the set of secondary joints.

### 8.1.1 Example: Five-Point Wheel Suspension

As an example of a non-explicitly solvable system, we regard a five-point wheel suspension of the rear-axis of a modern passenger car (in fact, the Daimler-Benz W201) (Fig. 33). A schematic model of the wheel suspension is displayed in Fig. 34.

For the description of the system, $n_k = 7$ reference systems were introduced, which will play the role of nodes in the associated graph. The reference frame $\mathcal{K}_1$ embodies the chassis, while the reference frame $\mathcal{K}_3$ represents the wheel carrier. The other five reference frames are attached to the rods. The bodies of the system are interconnected through five universal joints denoted "$U$" and five spherical joints denoted "$S$".

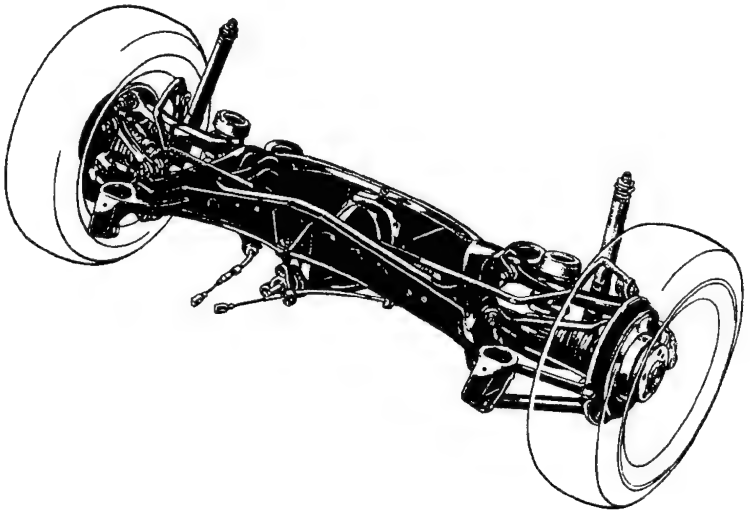


Figure 33: Rear axis of a Daimler-Benz W201

The number of independent loops of the mechanism is $n_L = n_E - n_K + 1 = 4$. A minimal cycle basis is shown in Fig. 34, where all loops contain the joints $U_1$ and $S_1$. For each loop, one can generate a kinematical transformer with has $f_{L_i} = 4$ local degrees of freedom. The coupling conditions between these kinematical transformers arise at joints $U_1$ and $S_1$, which can be regarded to be composed of five elementary revolute joints $\mathcal{G}_i$. For each of these, the number of coupling conditions is $p(\mathcal{G}_i) = n_L(\mathcal{G}_i) - n_B(\mathcal{G}_i) + 1 = 4 - 2 + 1 = 3$. Thus, the number of degrees of freedom of the complete subsystem is

$$f = \sum_{i=1}^{4} f_{L_i} - \sum_{i=1}^{5} p(\mathcal{G}_i) = 16 - 15 = 1 \quad ,$$

which corresponds to the result of the classical Gruebler-Kutzbach formula.

From the corresponding kinematical network it is not possible to recognize a recursive solution flow. Thus the algorithm switches to the iterative-solution mode, and proceeds to find an optimal spanning tree. As independent coordinate, one of the joints angles of $U_1$ is chosen. Next, a graph is produced which has as nodes the seven reference frames defined above and as edges the joint interconnections. Each edge is weighted by the number of degrees of freedom of the joints, which is three for the spherical joints and two for the universal joints (Fig. 35). Note that one universal joint is marked with a weight of only one as this is the joint from which the input variable was chosen.

The algorithm for finding the minimum-weight spanning tree starts with $T^{(0)} = \langle K^{(0)}, F^{(0)} \rangle = \langle \{\mathcal{K}_1\}, \{\} \rangle$. In the first application of step 2, the set of edges

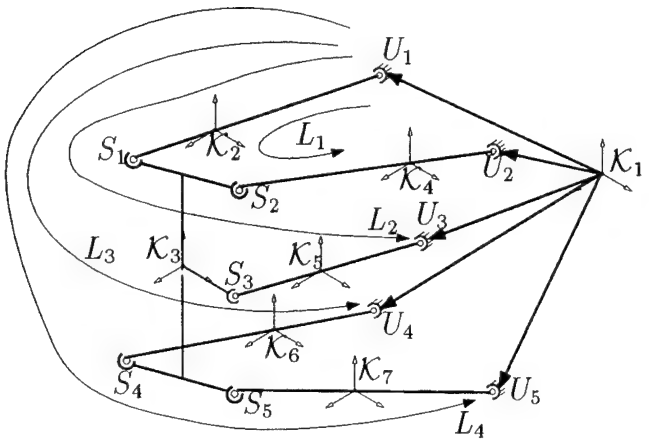



Figure 34: Schematic model of the five-point wheel suspension

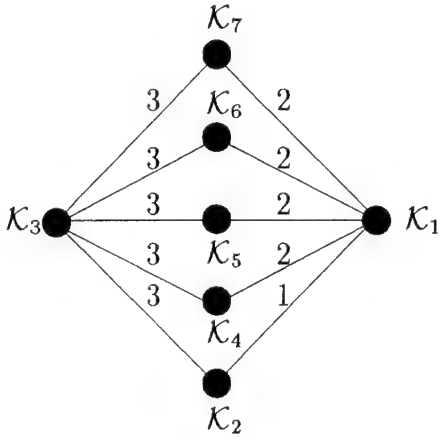



Figure 35: Interconnection graph for a five-point wheel suspension

$X^{(1)} = \{e_{12}, e_{14}, e_{15}, e_{16}, e_{17}\}$ is produced. From this set, the edge $e_{12}$ is identified as the shortest one. Therefore, $T^{(1)} = \langle\{\mathcal{K}_1, \mathcal{K}_2\}, \{e_{12}\}\rangle$ is computed. In the second application of step 2, the following sets are generated:

$$\begin{aligned} X^{(2)} &= \{e_{14}, e_{15}, e_{16}, e_{17}, e_{23}\} \ , \\ K^{(2)} &= \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_4\} \ , \\ F^{(2)} &= \{e_{12}, e_{14}\} \ . \end{aligned}$$

After $n_k - 1 = 6$ repetitions of step 2, one obtains the minimum-weight spanning tree as

$$T^{(6)} = \langle\{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_4, \mathcal{K}_5, \mathcal{K}_6, \mathcal{K}_7, \mathcal{K}_3\}, \{e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{23}\}\rangle \ .$$

The set of secondary joints is $C = \{e_{4,3}, e_{5,3}, e_{6,3}, e_{7,3}\}$. In this case, each of the secondary joints is a spherical one.

## 8.2 Formulation of Cut-Set Equations

After identifying the sets of secondary joints $C$, the corresponding constraint equations need to be formulated. Basically, each joint has associated with it a set of constraint equations. For a joint with $f_{\mathcal{G}_i}$ degrees of freedom, the number of these constraint equations is $n_c = 6 - f_{\mathcal{G}_i}$ for the spatial case and $n_c = 3 - f_{\mathcal{G}_i}$ for the planar or spherical case. The formulation of the constraint equations for different types of joints poses no fundamental problem. Here we describe the constraint equations for two types of joints which occur particularly often in the analysis of vehicle dynamics: (I) the spatial spherical joint and (II) the planar revolute joint. Further sets of constraint equations can be found for example in Nikravesh 1988 or Wehage and Janosi 1993.

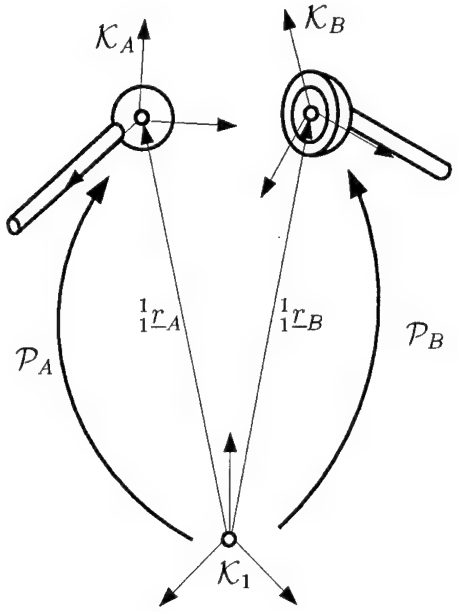



Figure 36: Constraint equations for a spherical joint

### 8.2.1 Constraint Equations for a Spherical Joint

The spherical joint involves $n_c = 6 - 3 = 3$ constraint equations for the spatial case. For their formulation, one introduces a reference system at each end of the spherical joint, which will be denoted as $\mathcal{K}_A$ and $\mathcal{K}_B$, respectively (Fig. 36). The constraint equations state that the position of the origins of the frames $\mathcal{K}_A$ and $\mathcal{K}_B$ with respect to a common frame $\mathcal{K}_1$ along two branches $\mathcal{P}_A$ and $\mathcal{P}_B$ must coincide. With the corresponding representations

$$ {}^1_1\underline{r}_A = \mathtt{Rot}[\,\mathrm{A}_{\mathcal{P}_A}]^T\,\mathtt{Trans}[\,\mathrm{A}_{\mathcal{P}_A}]\ , \tag{77} $$

$$ {}^1_1\underline{r}_B = \mathtt{Rot}[\,\mathrm{A}_{\mathcal{P}_B}]^T\,\mathtt{Trans}[\,\mathrm{A}_{\mathcal{P}_B}]\ , \tag{78} $$

the closure condition becomes

$$ {}^1_1\underline{r}_A - {}^1_1\underline{r}_B = \underline{0}\ . \tag{79} $$

Hereby, the branches $\mathcal{P}_A$ and $\mathcal{P}_B$ are known from the minimum weight spanning tree algorithm.

The corresponding linear velocity constraint equations result from

$$ {}^1_1\dot{\underline{r}}_A - {}^1_1\dot{\underline{r}}_B = \underline{0}\ , \tag{80} $$

with

$$ {}_1^1\dot{\underline{r}}_A = \mathtt{Rot}[\,\mathrm{A}_{\mathcal{P}_A}]^T\ {}_1^A\dot{\underline{r}}_A \quad , \quad {}_1^A\dot{\underline{r}}_A = \mathtt{Trans}[\,[{}^1\boldsymbol{J}_A]_{\mathcal{P}_A}]\ [\dot{\underline{\beta}}]_{\mathcal{P}_A} \quad , \tag{81} $$

$$ {}_1^1\dot{\underline{r}}_B = \mathtt{Rot}[\,\mathrm{A}_{\mathcal{P}_B}]^T\ {}_1^B\dot{\underline{r}}_B \quad , \quad {}_1^B\dot{\underline{r}}_B = \mathtt{Trans}[\,[{}^1\boldsymbol{J}_B]_{\mathcal{P}_B}]\ [\dot{\underline{\beta}}]_{\mathcal{P}_B} \quad , \tag{82} $$

where $\mathtt{Trans}[\,[{}^1\boldsymbol{J}_A]_{\mathcal{P}_A}]$ and $\mathtt{Trans}[\,[{}^1\boldsymbol{J}_B]_{\mathcal{P}_B}]$ represent the translational part, i.e., the last three rows, of the Jacobians associated with the two paths $\mathcal{P}_A$ and $\mathcal{P}_B$, respectively. Derivation of equation (80) with respect to time yields the acceleration constraint equation

$$ {}_1^1\ddot{\underline{r}}_A - {}_1^1\ddot{\underline{r}}_B = \underline{0} \quad , \tag{83} $$

with

$$ \left.\begin{array}{rcl} {}_1^1\ddot{\underline{r}}_A & = & \mathtt{Rot}[\,\mathrm{A}_{\mathcal{P}_A}]^T\ {}_1^A\ddot{\underline{r}}_A \\ {}_1^A\ddot{\underline{r}}_A & = & \mathtt{Trans}[\,[{}^1\boldsymbol{J}_A]_{\mathcal{P}_A}]\ [\ddot{\underline{\beta}}]_{\mathcal{P}_A} + \mathtt{Trans}[\,[{}^1\dot{\boldsymbol{J}}_A]_{\mathcal{P}_A}]\ [\dot{\underline{\beta}}]_{\mathcal{P}_A} \end{array}\right\} \quad , \tag{84} $$

$$ \left.\begin{array}{rcl} {}_1^1\ddot{\underline{r}}_B & = & \mathtt{Rot}[\,\mathrm{A}_{\mathcal{P}_B}]^T\ {}_1^B\ddot{\underline{r}}_B \\ {}_1^B\ddot{\underline{r}}_B & = & \mathtt{Trans}[\,[{}^1\boldsymbol{J}_B]_{\mathcal{P}_B}]\ [\ddot{\underline{\beta}}]_{\mathcal{P}_B} + \mathtt{Trans}[\,[{}^1\dot{\boldsymbol{J}}_B]_{\mathcal{P}_A}]\ [\dot{\underline{\beta}}]_{\mathcal{P}_A} \end{array}\right\} \quad . \tag{85} $$

### 8.2.2 Constraint Equations for the Planar Revolute Joint

The planar revolute joint results from the application of a spherical joint in a planar system. In this case, the number of constraint equations is $n_c = 3-1 = 2$. Supposing that the plane of motion is spanned by the basis vectors $\underline{e}_i$ and $\underline{e}_j$, and that the direction of the axis of the revolute joint is thus $\underline{e}_k = \underline{e}_i \times \underline{e}_j$, the two constraint equations can be written as

$$ {}_1^1\underline{r}_A\ [{}^1\underline{e}_i]_{\mathcal{P}_A} - {}_1^1\underline{r}_B\ [{}^1\underline{e}_i]_{\mathcal{P}_B} = 0 \quad , \tag{86} $$

$$ {}_1^1\underline{r}_A\ [{}^1\underline{e}_j]_{\mathcal{P}_A} - {}_1^1\underline{r}_B\ [{}^1\underline{e}_j]_{\mathcal{P}_B} = 0 \quad , \tag{87} $$

with

$$ \left.\begin{array}{ll} [{}^1\underline{e}_i]_{\mathcal{P}_A} = \mathtt{Rot}[\,\mathrm{A}_{\mathcal{P}_A}]\ \underline{e}_i \quad , & [{}^1\underline{e}_j]_{\mathcal{P}_A} = \mathtt{Rot}[\,\mathrm{A}_{\mathcal{P}_A}]\ \underline{e}_j \\ [{}^1\underline{e}_i]_{\mathcal{P}_B} = \mathtt{Rot}[\,\mathrm{A}_{\mathcal{P}_B}]\ \underline{e}_i \quad , & [{}^1\underline{e}_j]_{\mathcal{P}_B} = \mathtt{Rot}[\,\mathrm{A}_{\mathcal{P}_B}]\ \underline{e}_j \end{array}\right\} \quad . \tag{88} $$

The corresponding equations for velocities and accelerations are

$$ {}_1^1\dot{\underline{r}}_A\ [{}^1\underline{e}_i]_{\mathcal{P}_A} - {}_1^1\dot{\underline{r}}_B\ [{}^1\underline{e}_i]_{\mathcal{P}_B} = 0 \quad , \tag{89} $$

$$ {}_1^1\dot{\underline{r}}_A\ [{}^1\underline{e}_j]_{\mathcal{P}_A} - {}_1^1\dot{\underline{r}}_B\ [{}^1\underline{e}_j]_{\mathcal{P}_B} = 0 \quad , \tag{90} $$

and

$$ {}_1^1\ddot{\underline{r}}_A\ [{}^1\underline{e}_i]_{\mathcal{P}_A} - {}_1^1\ddot{\underline{r}}_B\ [{}^1\underline{e}_i]_{\mathcal{P}_B} = 0 \quad , \tag{91} $$

$$ {}_1^1\ddot{\underline{r}}_A\ [{}^1\underline{e}_j]_{\mathcal{P}_A} - {}_1^1\ddot{\underline{r}}_B\ [{}^1\underline{e}_j]_{\mathcal{P}_B} = 0 \quad . \tag{92} $$

# 9 Global Kinematics

As a final step of the kinematical processing, the absolute motion of the constituent bodies has to be calculated. To this end, one needs expressions for the relative motion at all joints and an arrangement of paths leading from the frame of reference to the bodies of the complete system. With the modules described so far, these issues have already been addressed on a cluster-by-cluster basis. This section discusses an overall procedure for amalgamating the derived results into a global processing module.

## 9.1 Global Representation of System Topology

In the preceding sections, several concepts were introduced related to the topological and geometrical processing of subsets of a general multibody system. For each of these concepts, a tailored topological system description was introduced that made it possible to formulate and solve the associated problems most efficiently. At this point, it is necessary to bring together the aforementioned results and to integrate them into an overall procedure. In order to ease this integration process, the topological information gathered so far was cast into a SOLVAS-compatible format, which is the format currently in use at the System Simulation & Technical Division Group at U.S. Army TARDEC. This format is based on a number of graph-theoretic concepts that were developed at U.S. Army TARDEC (Wehage and Belczynski 1993), and are explained below.

### 9.1.1 The Arc Connectivity Matrix $C_a$

The arc connectivity matrix describes the spanning-tree part of a graph. Each node of the graph, apart from the root, is associated with a column of this matrix. Moreover, each arc corresponds to a row. The matrix $C_a$ is thus a square matrix. An element of a row is `1` when the associated arc starts at the corresponding node, `-1` when the associated arc ends at the corresponding node, and `0` otherwise. As the root node does not appear in the matrix, all rows associated with arcs for which the start node is the root exhibit only one `1` apart from zeroes. By an appropriate permutation of columns and rows, $C_a$ can be made strictly lower triangular.

### 9.1.2 The Chord Connectivity Matrix $C_c$

The chord connectivity matrix describes the co-tree of the graph. Again, the columns are associated with the nodes of the graph, and the root node is not included in the matrix. Each row now corresponds a chord, i. e. an element of the row is `1` if the

column is associated with the start node of the chord, `-1` if this column corresponds to the end node of the chord, and `-0` otherwise. Note that $C_c$ is a rectangular matrix that describes also a set of independent loops, as each chord can be uniquely assigned to one loop of the system.

### 9.1.3 The Path-Tracing Matrix $R_a$

This matrix, defined in Wehage and Janosi 1993, is a square, strictly lower triangular matrix that results as the inverse of $C_a$. Each row describes the (unique) path from the root to the terminal node of one arc, without counting chord interconnections. The nodes which are part of the path are identified by a `1` in the corresponding column of the row, while all other elements are zero.

### 9.1.4 The Loop Closure Matrix $R_a$

This is the right inverse of the chord connectivity matrix, $R_c$, which can be computed as $R_c = -C_c\, R_a$ (Wehage and Janosi 1993). Each row of this matrix corresponds to the loop associated with a chord, the nonzero entries identifying the arcs which, together with the chord at hand, will form the loop. If, upon cycling in direction of the chord through the loop, the arc is traversed in its own sense, the entry is `1`, otherwise, i.e., if the arc is traversed in opposite sense, the entry is `-1`.

## 9.2 Generation of System-Topology Matrices

The path-tracing matrix $R_a$ and the chord connectivity matrix $C_c$ can be calculated directly from the internal path data generated during the decomposition and loop detection algorithms. Hereby, one proceeds cluster by cluster, beginning with the system reference frame, and taking as starting node for each cluster the end of the bridge that connects it to the preceding cluster. For each cluster, one has one of the following three cases:

1. Tree-type clusters. In this case, the matrix $C_c$ is empty, and the path-tracing matrix is obtained by traversing the bridges from node to node.

2. Non-recursively solvable clusters. Here, the path tracing matrix is just a copy of the minimum-weight spanning tree representation, and the chord connectivity matrix corresponds to the elements of the co-tree established in Section **8**.

3. Recursively solvable clusters. In this case, the path-tracing matrix is generated from the row of the shortest-path matrix $P$ described in Section **4**, which describes the shortest paths from the local root to all other nodes of the

subsystem. Matrix $C_c$ results as the set of edges not contained in this row of $P$.

After determining $C_c$ and $R_a$, the other two matrices, $C_a$ and $R_c$, are established as the inverse of $R_a$ and $R_c = -C_c\, R_a$, respectively.

## 9.3 Generation of Absolute Kinematics

After establishing the four matrices described above, the generation of the absolute kinematics consists in tracking the paths between the bodies of the system and the global reference frame and superposing the known expressions for the relative kinematics at the corresponding arcs using the kinematical expressions for serial chains described in Section **2.3**. Hereby, one can employ the information stored in the path-tracing matrix $R_a$, re-using intermediate expressions by traversing the spanning tree in a width-first manner.

## 9.4 Example: Wheel Suspension of a Trailer

The generation of the SOLVAS-compatible topological system representation matrices as well as the expressions for the absolute kinematics shall be illustrated for the wheel suspension described in Section **2.6.2**. The corresponding graph is displayed in Fig. 37, where the arcs are represented as full lines and the chord is represented as a dashed line. Note that this system is recursively solvable, so the arcs result from the shortest-path matrix generated with the methods of Section **4**. The resulting connectivity and path matrices are thus

$$C_a = \begin{array}{c} \begin{matrix} \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 & \mathcal{K}_6 & \mathcal{K}_7 & \mathcal{K}_8 \end{matrix} \\ \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{array}\right] \begin{array}{c} e_{12} \\ e_{13} \\ e_{24} \\ e_{45} \\ e_{36} \\ e_{67} \\ e_{48} \end{array} \end{array} \quad , \quad R_a = \begin{array}{c} \begin{matrix} \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 & \mathcal{K}_6 & \mathcal{K}_7 & \mathcal{K}_8 \end{matrix} \\ \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \begin{array}{c} e_{12} \\ e_{13} \\ e_{24} \\ e_{45} \\ e_{36} \\ e_{67} \\ e_{48} \end{array} \end{array} ,$$

$$C_c = \begin{array}{c} \begin{matrix} \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 & \mathcal{K}_6 & \mathcal{K}_7 & \mathcal{K}_8 \end{matrix} \\ \left[\begin{array}{ccccccc} 0 & 0 & 0 & -1 & 0 & 1 & 0 \end{array}\right] \begin{array}{c} e_{57} \end{array} \end{array} \quad , \quad R_c = \begin{array}{c} \begin{matrix} \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 & \mathcal{K}_6 & \mathcal{K}_7 & \mathcal{K}_8 \end{matrix} \\ \left[\begin{array}{ccccccc} 1 & -1 & 1 & 1 & -1 & -1 & 0 \end{array}\right] \begin{array}{c} e_{57} \end{array} \end{array} .$$

Hereby, the frames $\mathcal{K}_i$ act as nodes, while the edges $e_{ij}$ correspond to the transformations between the reference frames.

With these matrices, the absolute kinematics can be generated by issuing the following command

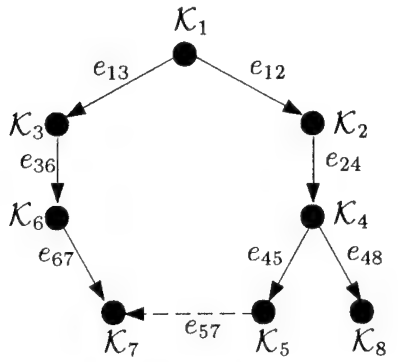



Figure 37: Topological structure of the trailer wheel suspension

```
GenerateGlobalKinematics{K,{s,alpha,beta,gamma},{s},{6,7,8}]
```

Here, `K` is the frame interconnectivity matrix defined in Section **2.6.2**, `{alpha,beta,gamma}` is the list of dependent joint variables (which have been supplied by the user as variables of the corresponding joints), `{s}` is the list of independent variables (again supplied by the user), and `{6,7,8}` is the list of indices of reference frames for which the absolute motion is desired to be produced. The ensuing code produced by the program is

```
R6[1, 1] -> 1;   R6[1, 2] -> 0;            R6[1, 3] -> 0
R6[2, 1] -> 0;   R6[2, 2] -> cos[beta];    R6[2, 3] -> -sin[beta]
R6[3, 1] -> 0;   R6[3, 2] -> sin[beta];    R6[3, 3] -> cos[beta]
R7[1, 1] -> 1;   R7[1, 2] -> 0;            R7[1, 3] -> 0
R7[2, 1] -> 0;   R7[2, 2] -> cos[beta];    R7[2, 3] -> -sin[beta]
R7[3, 1] -> 0;   R7[3, 2] -> sin[beta];    R7[3, 3] -> cos[beta]
R8[1, 1] -> 1;   R8[1, 2] -> 0;            R8[1, 3] -> 0
R8[2, 1] -> 0;   R8[2, 2] -> cos[alpha];   R8[2, 3] -> -sin[alpha]
R8[3, 1] -> 0;   R8[3, 2] -> sin[alpha];   R8[3, 3] -> cos[alpha]
r6[1] -> 0
r6[2] -> v2y cos[beta] - v2z sin[beta]
r6[3] -> -(v2z cos[beta]) - v2y sin[beta]
r7[1] -> 0
r7[2] -> ADH18[2, 4] cos[beta] + ADH7[3, 4] sin[beta]
r7[3] -> ADH7[3, 4] cos[beta] - ADH18[2, 4] sin[beta]
r8[1] -> 0
r8[2] -> ADH19[2, 4] cos[alpha] + ADH20[3, 4] sin[alpha]
r8[3] -> ADH20[3, 4] cos[alpha] - ADH19[2, 4] sin[alpha]
omega6[1] -> beta';    omega6[2] -> 0;   omega6[3] -> 0
omega7[1] -> beta';    omega7[2] -> 0;   omega7[3] -> 0
omega8[1] -> alpha';   omega8[2] -> 0;   omega8[3] -> 0
v6[1] -> 0;   v6[2] -> 0;          v6[3] -> 0
v7[1] -> 0;   v7[2] -> s beta';    v7[3] -> -s'
v8[1] -> 0;   v8[2] -> 0;          v8[3] -> -(v4y alpha')
```

```
JACR17[6] -> s beta'
JACR20[5] -> v4y alpha'
JACR19[5] -> 2 beta' s'
JACR19[6] -> JACR17[6] beta'
JACR21[5] -> JACR20[5] alpha'
alpha6[1] -> beta'';   alpha6[2] -> 0;   alpha6[3] -> 0
alpha7[1] -> beta'';   alpha7[2] -> 0;   alpha7[3] -> 0
alpha8[1] -> alpha'';   alpha8[2] -> 0;   alpha8[3] -> 0
a6[1] -> 0;   a6[2] -> 0;                      a6[3] -> 0
a7[1] -> 0;   a7[2] -> JACR19[5] + s beta'';   a7[3] -> JACR19[6] - s''
a8[1] -> 0;   a8[2] -> JACR21[5];              a8[3] -> -(v4y alpha'')
```

Here, it is assumed that the relative kinematics have already been solved, i.e., that the functions $\alpha(s), \beta(s)$ and $\gamma(s)$ as well as their first and second derivatives $\alpha' = \mathrm{d}\alpha(s)/\mathrm{d}s$, $\alpha'' = \mathrm{d}^2\alpha(s)/\mathrm{d}s^2$, etc. with respect to $s$ are known. The corresponding expressions are calculated during the processing of the relative kinematics and are not re-substituted here in order to avoid redundant multiplications. Note that terms `sin[beta]` and `cos[beta]` appear repeatedly in the expressions. At first sight, this might look like a redundant repeated evaluation of the computationally expensive trigonometric functions. However, this is not the case, as the actual functions of Mathematica are spelled out with the first letter in upper case. In fact, the aforementioned terms represent just constants that are calculated exactly *once* in the program and then used repeatedly in the resulting expressions.

# 10 Overview of the Implementation

The described methods were implemented using the symbolic programming language Mathematica. The resulting program, named SYMKIN, has about 4700 lines of code. The program is designed such as to perform all operations fully automatically. It is organised in nine modules, each of which is dedicated to one of the main processing stages of the algorithm developed above. These nine stages stem from the gradual decomposition of the system into subsystems which either are solvable in closed form or must be treated iteratively. For better clarity, this gradual decomposition and processing operations is summarized next based on the example of the heavy-load manipulator introduced in Section **2.6.3** and depicted in Fig. 38.

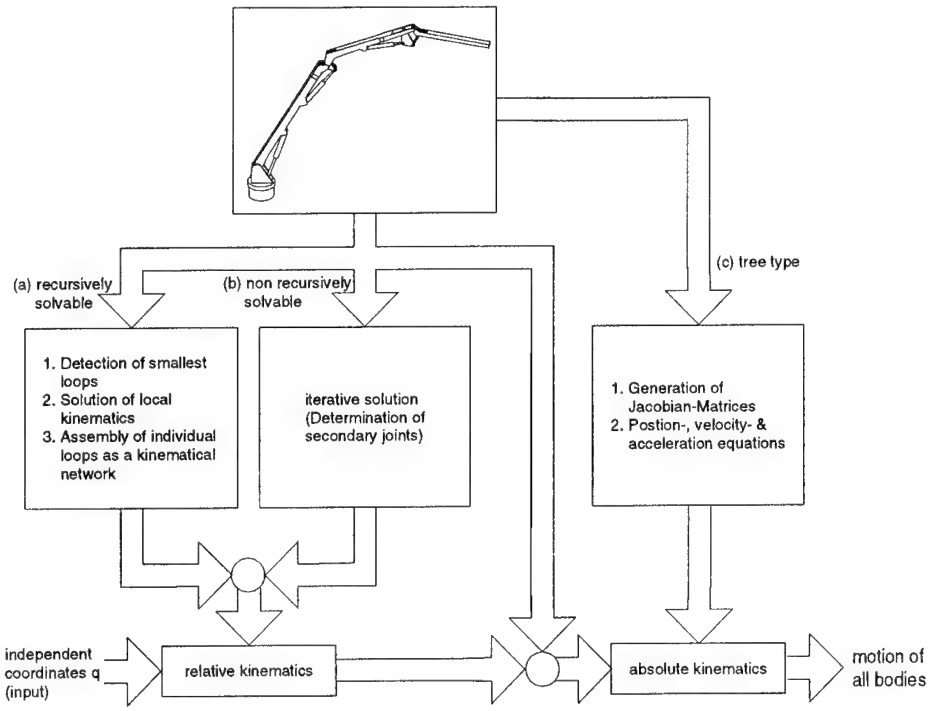



Figure 38: Overview of the modules of the SYMKIN package

Starting from a general system, a kinematic description is established and brought to normal form (Module 1). From this system, a set of subsystems is determined in which the bodies either are connected as loops, or form tree-type structures (Module 2). For each of the subsystems containing loops, which are termed "clusters", a set of smallest independent loops is established (Module 3). Then, a corresponding loop interconnection scheme, termed the "kinematical network", is computed, and it is decided whether the kinematics of the interconnected set of loops is recursively solvable or not (Module 4). If the kinematics of the multiple-loop subsystem are solvable in closed form, each of the loops is further processed and

a set of local closed-form solutions is generated for it, covering position, velocity, and acceleration (Module 5); then, the so-established local loop solutions are assembled together using the information of the kinematical network (Module 6). If the multiple-loop subsystem is not solvable in closed form, a set of constraint equations for their iterative solution is produced, and corresponding equations for velocities and accelerations are generated (Module 7). After generating in this way the local solutions for each cluster, the characteristic matrices for establishing the interconnection structure of the overall system are determined (Module 8). Finally, based on this information, and the information produced in the previous modules, the kinematical expressions for the absolute motion of all bodies is generated (Module 9). The hierarchical decomposition of the clusters, together with the different types of kinematical processing, are illustrated in Fig. 39.

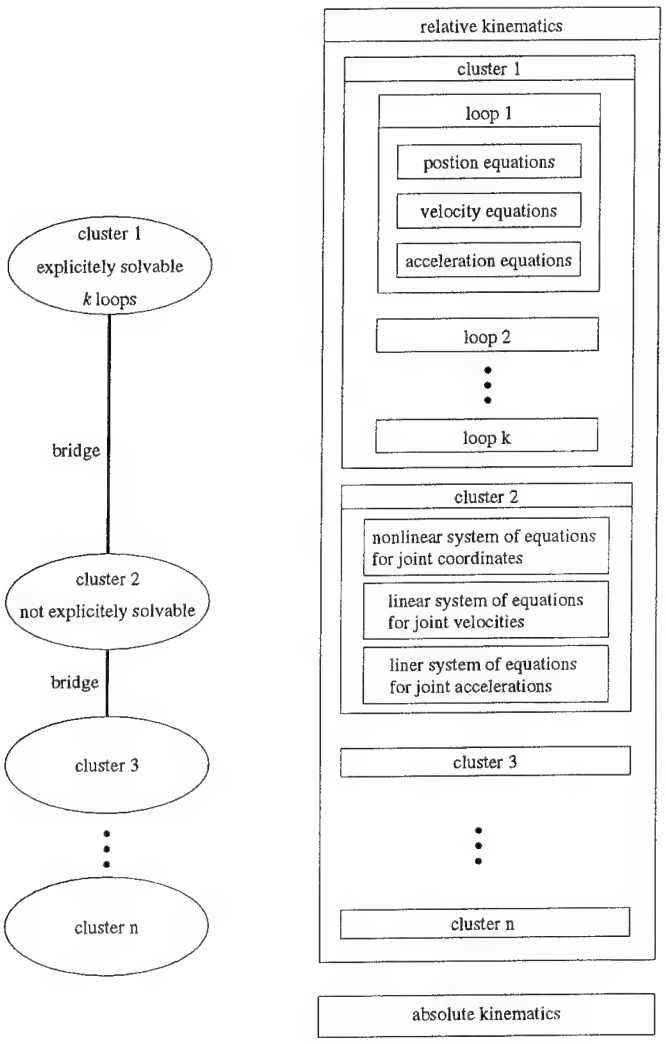



Figure 39: Hierarchical decomposition of a complex multibody system

## 10.1 Main Functions of the Package

The Mathematica package developed during this project has over 120 functions. A detailed description of all functions is thus beyond the scope of this report. In order to ease the use of the code, these operations were grouped in 15 mayor function calls through which the user can produce the desired sets of equations. These functions shall be described below (see also the tables (5), (6), (7)), (8)) and (9)).

The main functions basically mimic the main processing stages of the algorithm and are organized as follows:

**0.** System Description Stage.
The basic information that the user needs to supply are the frame connectivity matrix `K`, the set of joint variables, and a set of user-desired independent joint variables. The set of joint variables marks the parameters in the transformation matrices that are to be treated as non-constant quantities, i. e., for which velocity and acceleration are to be taken into account. The second set of variables represents the independent variables that the user wants to be employed as inputs to the system. This set does not have to be complete, i. e., the user can supply only a subset of input variables and leave the determination of the remaining independent variables to the program. In particular, the user-supplied set of independent variables can be empty, meaning that all inputs are to be established automatically. In order to build the frame connectivity matrix, one needs to dissect the multibody system into a set of elementary components, which can be joints, links, or assemblies hereof, and to mark the endpoints of these components by reference frames. The columns and rows of the matrix then correspond to the different reference frames of the system. The user supplies those elements of the matrix for which an interconnection between two neighboring nodes exists, prescribing the transformation sequence from the node associated with the corresponding row to the node associated to the corresponding column. Most of the elements of this matrix are thus empty. The model of the frame connectivity matrix `K` and the set of allowed transformations are described in Section **2**.

**1.** Parameter Normalization.
Non-elementary transformations like e.g. the Denavit-Hartenberg parameters and other composite transformations need to be reduced to sequences of elementary transformations in order for the algorithms to work properly. Normally, the normalization is carried automatically when one of the global processing functions described below are invoked. However, the user can carry out this normalization explicitly by invoking the function `SimplifyChain[...]`.

**2.** Detection of Clusters.
With the function `FindLeaves[...]`, the independent clusters of the system

are recognized and returned. As input, the adjacency matrix for the complete system, as well as the set of joint variables and the desired input coordinates are provided. Return value is an adjacency matrix and a set of joint variables for each cluster.

3. Generation of Independent Loops.
`GetLoopBasis[...]` generates a minimum cycle basis for one cluster, returning the set of cycles as the sequences of transformations and the sets of nodes that form one loop, respectively. The input for this function is the adjacency matrix for one cluster, as well as the set of joint variables and the desired input coordinates, as described above.

4. Determination of Kinematical Networks.
The function `GenerateNetwork[...]` recognises the couplings between the independent loops and generates the corresponding *kinematical network*. From this, it is checked whether there exists a recursive solution flow for the kinematical network. If this is the case, a list of input variables representing the recursive solution flow is generated for each loop. If not, the subsystem is marked as "non-recursively solvable" for subsequent steps. The input for this function is a set of independent loops and the set of joint variables and the desired input coordinates, as described above.

5. Closed-Form Loop Solution.
The function `SolveSingleLoop[...]` generates closed form solutions for each single loop, taking as input a sequence of transformations, as well as the set of joint variables and the desired input coordinates, as described above. The generation of the local loop kinematics occurs in several steps: (1) `GenerateConstraints[...]`, produces the set of scalar equations that can be solved in closed form, and determines the corresponding expressions for the coefficients $A, B, C$. Then, `SolvePositionEquations[...]` resolves these equations, producing either two solutions or one unique solution per equation, as applicable. Finally, expressions for the first and second time derivatives of the dependent joint variables are determined via `GenerateVelocities[...]` and `GenerateAccelerations[...]`. If the algorithm fails to find such a closed-form solution, it stops again, and the loop is termed "non-recursively-solvable".

6. Generation of implicit solutions.
For subsystems which are not explicitly solvable, a minimal set of constraint equations needs to be generated. This is done by `GenerateImplicitEquations[...]`. The function takes as inputs the adjacency matrix of a cluster, a set of transformation sequences of the loops representing the independent loops, as well as the set of joint variables and the desired input coordinates, as described above. Passing the set of trans-

formation sequences allows the function to recognize planar or spherical subsystems within the cluster and to generate the appropriate constraint equations. The said function returns a sequence of constructs termed *Numerical Solution Forms* or `NSF[substitutions,equations,coordinates]`, where `equations` denotes the set of implicit constraint equations to be solved iteratively, `coordinates` are the dependent variables, and `substitutions` represents a set of intermediate-value substitutions that arise in `equations`. Each numerical solution form corresponds to one scalar equation, which in general is implicit in the unknowns `coordinates`. The equation can be either nonlinear for the case of the position equations, or linear for the case of the velocity and acceleration equations.

7. Merging of Equations.
   This function, termed `MergeEquations[...]` merges all global equations and reorders them such that all terms appearing on the right side of any assignment have been defined in previous steps. The input for this function are the equation for the joint coordinates and their first and second derivative which were generated by `SolveSingleLoop[...]`.

8. Generation of SOLVAS-Matrices.
   The function `GetTopologicalStructure[...]` generates matrix-representations of the system topology that are compatible with the input format for the SOLVAS preprocessor of the System Simulation & Technical Division Group at U.S. Army TARDEC. The input for this function is just the global node adjacency matrix defined in stage 0. The generated matrices are the node connectivity matrices, the loop matrix for the minimal-cycle basis, as well as the matrix of shortest paths from the base to the bodies of the system. $C_a$ and $R_a$ are brought to lower triangular form by an appropriate column and row pivoting. The corresponding permutations of columns is stored in a list containing, for each column, the reference frame associated with it.

9. Generation of Absolute Kinematics
   With the function `SolveAbsoluteKinematics[...]`, the absolute orientation, position, velocity and acceleration of a set of specified reference frames can be computed. Hereby, the paths from the base to all bodies are taken according to the information stored in $R_a$. The function takes as inputs the global node adjacency matrix, the shortest-path matrix $R_a$, the column-permutation list defined in the previous stages, the list of frames of interest, as well as the set of joint variables and the desired input coordinates, as described above. As a result, it returns a sequence of equations for the absolute kinematics.

Apart from these functions, there are a number of convenience functions which allow to perform several steps at once or produce some simplifications of the resulting equations. These are

- ⋆ `GenerateGlobalKinematics`. This function produces the complete kinematics, including relative kinematics, closed-form and iterative solutions, and absolute kinematics.

- ⋆ `SolveRelativeKinematics`. This function determines the relative motion at all joints of a multiple-loop cluster in terms of the independent joint variables and their their first and second time-derivatives.

- ⋆ `GetSubstitutions`. This function generates a set of assignments for the intermediate expressions generated so far by the equation processing algorithms.

An overview of the function invocation hierarchy is depicted in Fig. 40.

| parameter | meaning |
|---|---|
| `K` | adjacency matrix |
| `jointVars` | list of joint variables |
| `inputVars` | set of user-desired input variables |

Table 5: Basic input parameters for kinematic processing modules

| function invokation | return parameters |
|---|---|
| `leafs = FindLeafs[K,jointVars, inputVars]` | `leafs[[1]]` : list of adjacency matrices for the leafs<br>`leafs[[2]]` : list of joint variables for the leafs<br>`leafs[[3]]` : list of input variables for the leafs |
| `loops = GetLoopBasis[K,jointVars, inputVars]` | `loops[[1]]` : list of transformation sequences fo the loops<br>`loops[[2]]` : sets of nodes for the loops |
| `solutionflow = GenerateNetwork[ loops[[1]], jointVars,inputVars]` | `solutionflow` : List of inputs for single loop kinematics |
| `ts = GetTopologicalStructure[K]` | `ts[[1]]` : list showing the permutation of columns<br>`ts[[2]]` : $R_a$<br>`ts[[3]]` : $C_a$<br>`ts[[4]]` : $R_c$<br>`ts[[5]]` : $C_c$ |

Table 6: Main routines for the topological processing

| function invokation | input and return parameters |
|---|---|
| `slk = SolveSingleLoop[loop,jointVars,inputVars]` | `loop` : transformation sequence<br>`slk` : equations for dependent joint coordinates and their derivatives |
| `con = GenerateConstraints[{loop,jointVars,inputVars}]` | `con[[1]]` : list of explicit solution forms<br>`con[[2]]` : internal data |
| `pos = SolvePositionEquations[con,jointVars,inputVars]` | `pos[[1]]` : equations for dependent joint coordinates<br>`pos[[2]]` : internal data |
| `vel = GenerateVelocities[jointVars,inputVars,pos[[2]]` | `vel[[1]]` : equations for dependent joint velocities<br>`vel[[2]]` : internal data |
| `acc = GenerateAccelerations[jointVars,inputVars, vel[[2]],pos[[2]]]` | `acc` : equations for dependent joint accelerations |

Table 7: Routines for the treatment of single loops



| function invokation | return parameters |
|---|---|
| `SolveRelativeKinematics[K,jointVars,inputVars]` | equations for the complete relative kinematics of the system |
| `GenerateImplicitSolutions[K,loops[[1]], jointVars,inputVars` | numerical solution forms |

Table 8: Routines for the treatment of clusters

| function invokation | input and return parameters |
|---|---|
| `ak = SolveAbsoluteKinematics[K, jointVars,inputVars, ts[[1]],ts[[2]],Frames]` | `Frames` : set of frames for which the absolute kinematics should be processed<br>`ak` : sequence of equations for the absolute kinematics |
| `gk = GenerateGlobalKinematics[K, jointVars,inputVars,Frames]` | `gk` : sequence of equations for the global kinematics |

Table 9: Routines for global processing

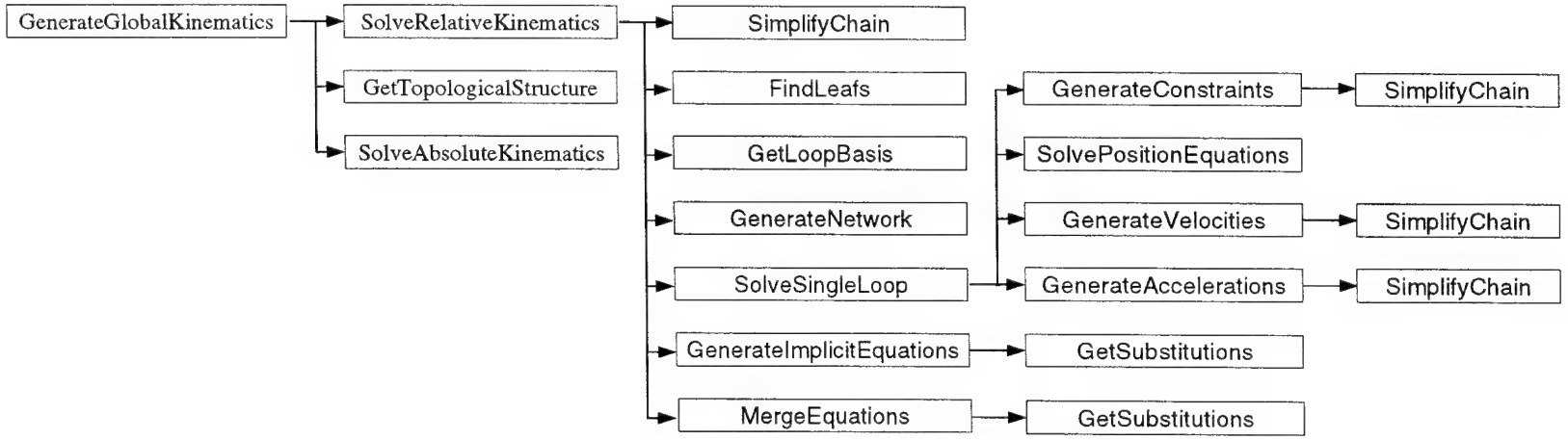



Figure 40: Function invokation hierachy of the SYMKIN package

## 10.2 Example of a Complete Kinematics Processing Session

The discussed procedures where applied to the example of the wheel suspension of a trailer described in Section **2.6.2**. Shown in Fig. 41 is the result of the program after issuing the command "`GenerateGlobalKinematics`". As it can be seen, the complete kinematics for this system fit into one sheet of paper.

```
In[1]:= << GlobalKinematics.m
In[2]:=  K=Table[Null,{8},{8}];
In[3]:=  K[[1,2]] = ETransform[2,1,v1y];
In[4]:=  K[[1,3]] = ETransform[2,1,v2y].ETransform[3,1,-v2z];
In[5]:=  K[[2,4]] = ETransform[1,0,alpha];
In[6]:=  K[[4,5]] = ETransform[2,1,-v3y].ETransform[3,1,v3z];
In[7]:=  K[[3,6]] = ETransform[1,0,beta];
In[8]:=  K[[7,6]] = ETransform[3,1,s];
In[9]:=  K[[5,7]] = ETransform[1,0,gamma];
In[10]:= K[[4,8]] = ETransform[2,1,-v4y];
In[11]:= Joints      = {s,alpha,beta,gamma}
In[12]:= InputJoints = {s};
In[13]:= GenerateGlobalKinematics[K,Joints,InputJoints,{6,7,8}];
In[14]:= % //TableForm
Out[14]:= //TableForm=
    ADH1[2, 4] -> -v1y + v2y
                   2
    ADH3[1, 1] -> s
                  2     2
    ADH5[1, 1] -> v3y  + v3z
                  2              2
    ADH2[1, 1] -> v2z  + ADH1[2, 4]
    beta -> -ArcTan[Sqrt[-1 +
                      2     2             2
                   4 s  (v2z  + ADH1[2, 4] )
    >       ----------------------------------------]] +
                                                    2
            (ADH2[1, 1] + ADH3[1, 1] - ADH5[1, 1])
    >       ArcTan[-2 s v2z, -2 s ADH1[2, 4]]
    ADH8[2, 4] -> s Sin[beta]
    ADH8[3, 4] -> -(s Cos[beta])
    ADH6[2, 4] -> -v1y + v2y + ADH8[2, 4]
    ADH7[3, 4] -> -v2z + ADH8[3, 4]
                  -(v3y ADH6[2, 4]) + v3z ADH7[3, 4]
    alpha -> ArcTan[-----------------------------------,
                                2     2
                              v3y  + v3z
    ADH14[2, 4] -> v3y Cos[gamma] - v3z Sin[gamma]
    ADH14[3, 4] -> -(v3z Cos[gamma]) - v3y Sin[gamma]
                  s'
    alpha' -> -----------
              ADH14[2, 4]
             ADH14[3, 4] alpha'
    beta' -> ------------------
                    s
    gamma' -> -alpha' + beta'
    JACR6[6] -> -(s beta')
    JACR8[5] -> -(ADH14[2, 4] alpha')
    JACR8[6] -> -(ADH14[3, 4] alpha')
    JACR9[5] -> -(ADH14[2, 4] alpha') - ADH14[2, 4] gamma'
    JACR9[6] -> -(ADH14[3, 4] alpha') - ADH14[3, 4] gamma'
    JACR10[5] -> -(JACR9[5] alpha') - JACR8[5] gamma'
    JACR10[6] -> -(JACR9[6] alpha') - JACR8[6] gamma'
    JACR7[6] -> -(JACR6[6] beta')
                -JACR10[6] + JACR7[6] + s''
    alpha'' -> ---------------------------
                      ADH14[2, 4]
              -JACR10[5] + ADH14[3, 4] alpha''
    beta'' -> --------------------------------
                            s
    gamma'' -> -alpha'' + beta''
    R6[1, 1] -> 1
    R6[1, 2] -> 0
    R6[1, 3] -> 0
    R6[2, 1] -> 0
    R6[2, 2] -> Cos[beta]
    R6[2, 3] -> -Sin[beta]
    R6[3, 1] -> 0
    R6[3, 2] -> Sin[beta]
    R6[3, 3] -> Cos[beta]
    R7[1, 1] -> 1
    R7[1, 2] -> 0
    R7[1, 3] -> 0
    R7[2, 1] -> 0
    R7[2, 2] -> Cos[beta]
    R7[2, 3] -> -Sin[beta]
    R7[3, 1] -> 0
    R7[3, 2] -> Sin[beta]
    R7[3, 3] -> Cos[beta]
    R8[1, 1] -> 1
    R8[1, 2] -> 0
    R8[1, 3] -> 0
    R8[2, 1] -> 0
    R8[2, 2] -> Cos[alpha]
    R8[2, 3] -> -Sin[alpha]
    R8[3, 1] -> 0
    R8[3, 2] -> Sin[alpha]
    R8[3, 3] -> Cos[alpha]
    r6[1] -> 0
    r6[2] -> v2y Cos[beta] - v2z Sin[beta]
    r6[3] -> -(v2z Cos[beta]) - v2y Sin[beta]
    r7[1] -> 0
    r7[2] -> ADH18[2, 4] Cos[beta] + ADH7[3, 4] Sin[beta]
    r7[3] -> ADH7[3, 4] Cos[beta] - ADH18[2, 4] Sin[beta]
    r8[1] -> 0
    r8[2] -> ADH19[2, 4] Cos[alpha] + ADH20[3, 4] Sin[alpha]
    r8[3] -> ADH20[3, 4] Cos[alpha] - ADH19[2, 4] Sin[alpha]
    omega6[1] -> beta'
    omega6[2] -> 0
    omega6[3] -> 0
    omega7[1] -> beta'
    omega7[2] -> 0   omega7[3] -> 0
    omega8[1] -> alpha'
    omega8[2] -> 0
    omega8[3] -> 0
    v6[1] -> 0
    v6[2] -> 0
    v6[3] -> 0
    v7[1] -> 0
    v7[2] -> s beta'
    v7[3] -> -s'
    v8[1] -> 0
    v8[2] -> 0
    v8[3] -> -(v4y alpha')
    JACR17[6] -> s beta'
    JACR20[5] -> v4y alpha'
    JACR19[5] -> 2 beta' s'
    JACR19[6] -> JACR17[6] beta'
    JACR21[5] -> JACR20[5] alpha'
    alpha6[1] -> beta''
    alpha6[2] -> 0
    alpha6[3] -> 0
    alpha7[1] -> beta''
    alpha7[2] -> 0
    alpha7[3] -> 0
    alpha8[1] -> alpha''
    alpha8[2] -> 0
    alpha8[3] -> 0
    a6[1] -> 0
    a6[2] -> 0
    a6[3] -> 0
    a7[1] -> 0
    a7[2] -> JACR19[5] + s beta''
    a7[3] -> JACR19[6] - s''
    a8[1] -> 0
    a8[2] -> JACR21[5]
    a8[3] -> -(v4y alpha'')
```

Figure 41: Mathematica-session for the trailer wheel suspension

# 11 Example: A Comprehensive Mixing Unit of the Control Mechanism of a Helicopter

As an example of a complex mechanical system, we regard in the following a subunit of the control mechanism of the main rotor of the helicopter "BO 105" of the German manufacturer Messerschmitt-Bölkow-Blohm MBB shown in Fig. 42.

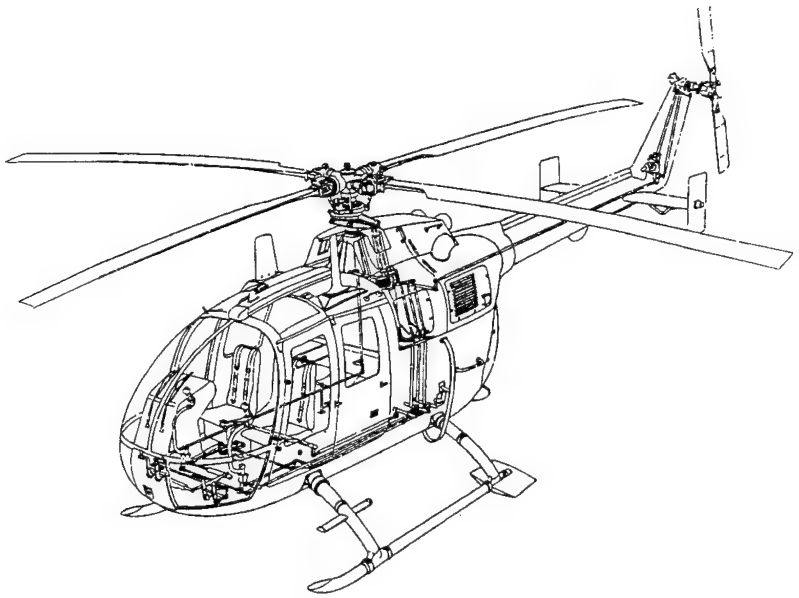


Figure 42: Helicopter Messerschmidt-Bölkow-Blohm BO 105

The task of the control mechanism is to transmit the actuation of the steering levers in the cockpit to the orientation of the rotor blades, as shown in Fig. 43. The actuations are performed in one of three ways: a longitudinal actuation $\Theta_{Lo}$ and a lateral actuation $\Theta_{La}$ of the cyclic stick, and a vertical actuation $\Theta_C$ of the collective pitch lever. The control motions of the pilot are transmitted by four-bar mechanisms and amplified in an hydraulic unit. From there they are further transmitted to the mixer unit, where the three control motions are superposed and brought to the fixed part of the swash plate. The attitude of the swash plate effects a cyclic motion of the rotating control rods that finally realize the desired angle of aviation $\gamma$ of the blades a a function of main rotor rotation angle $\alpha$.

The subunit regarded here is the mixer unit, which is depicted again in Fig. 44 for easier reference. The mechanism consists of 16 bodies, including the base body, which are interconnected by 21 joints. The corresponding joint variables are displayed in Table 10. The overall degree of freedom of the unit is $f = 3$. In the following, the generation of the complete kinematics, i.e., the position, velocity,

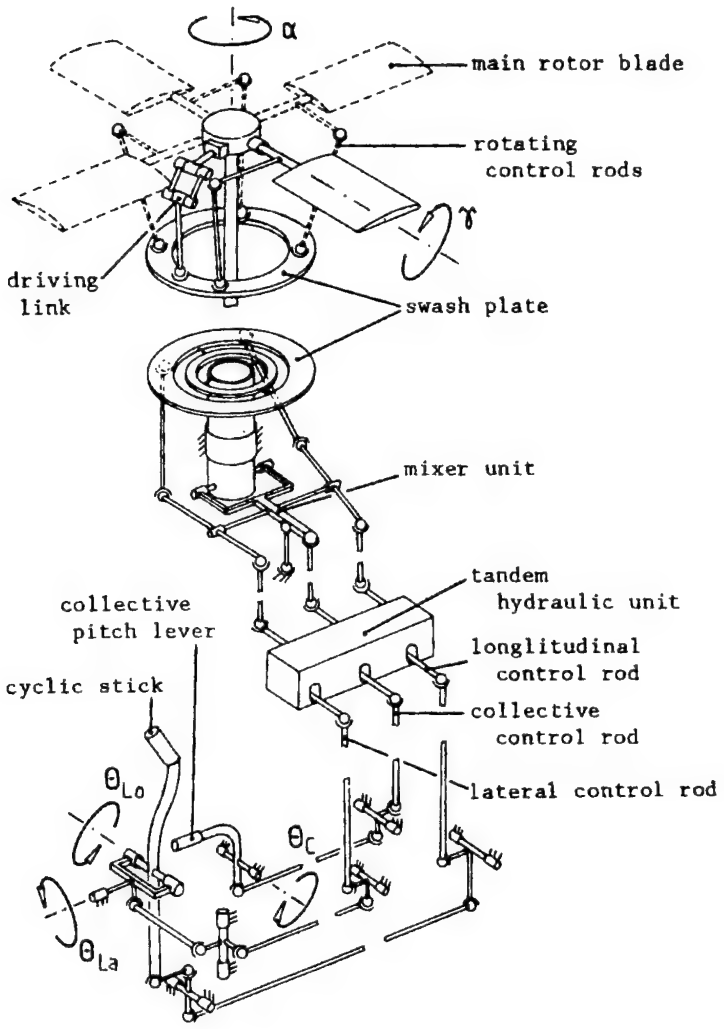



Figure 43: Control mechanism of the main rotor

and acceleration transmission functions, for the mechanism will be regarded. The choice of the independent variables as well as the selection of an appropriate set of independent loops will be left to the algorithm.

In order to state the topological description of the system, we need an appropriate frame adjacency matrix $K$, for which we associate with each body one reference frame. These associations are depicted in Fig. 45, where the actual positions of the frames are not displayed for better clarity. For example, $\mathcal{K}_1$ describes the base body, while $\mathcal{K}_2$, $\mathcal{K}_3$ and $\mathcal{K}_4$ represent the lateral, collective, and longitudinal control of the mechanism, respectively. The non-rotating part of the swash plate is represented by $\mathcal{K}_{16}$ and the non-rotating control rods by $\mathcal{K}_{14}$ and $\mathcal{K}_{15}$. Apart from the reference frames, Fig. 46 shows also the link lengths and the joint denominations. According

to this, and together with the symbols of Table 10, the input file for the SYMKIN package takes the form depicted in Fig. 46. Note that no joint coordinates are specified as input coordinates, so that the program must select appropriate ones.

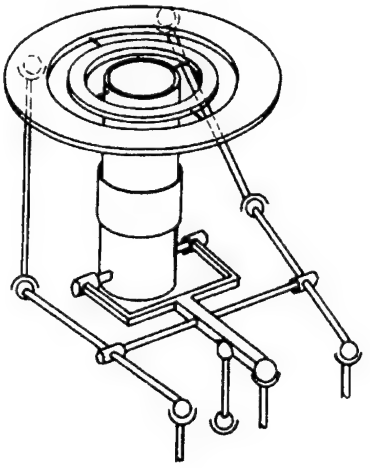


Figure 44: Mixer unit of the control mechanism of the main rotor

| joint | joint coordinates | type | joint | joint coordinates | type |
|---|---|---|---|---|---|
| $R_1$ | $\delta_1$ | revolute | $P_1$ | $s_1$ | prismatic |
| $R_2$ | $\delta_2$ | revolute | $P_2$ | $s_2$ | prismatic |
| $R_3$ | $\delta_3$ | revolute | $P_3$ | $s_3$ | prismatic |
| $R_4$ | $\delta_4$ | revolute | $P_4$ | $s_4$ | prismatic |
| $R_5$ | $\delta_5$ | revolute | | | |
| $T_1$ | $\beta_1, \beta_2$ | universal | $S_1$ | $\phi_1, \phi_2, \phi_3$ | spherical |
| $T_2$ | $\beta_3, \beta_4$ | universal | $S_2$ | $\phi_4, \phi_5, \phi_6$ | spherical |
| $T_3$ | $\beta_5, \beta_6$ | universal | $S_3$ | $\phi_7, \phi_8, \phi_9$ | spherical |
| $T_4$ | $\beta_7, \beta_8$ | universal | $S_4$ | $\phi_{10}, \phi_{11}, \phi_{12}$ | spherical |
| $T_5$ | $\beta_9, \beta_{10}$ | universal | $S_5$ | $\phi_{13}, \phi_{14}, \phi_{15}$ | spherical |
| $T_6$ | $\beta_{11}, \beta_{12}$ | universal | $S_6$ | $\phi_{16}, \phi_{17}, \phi_{18}$ | spherical |

Table 10: Joints of the helicopter mixer unit

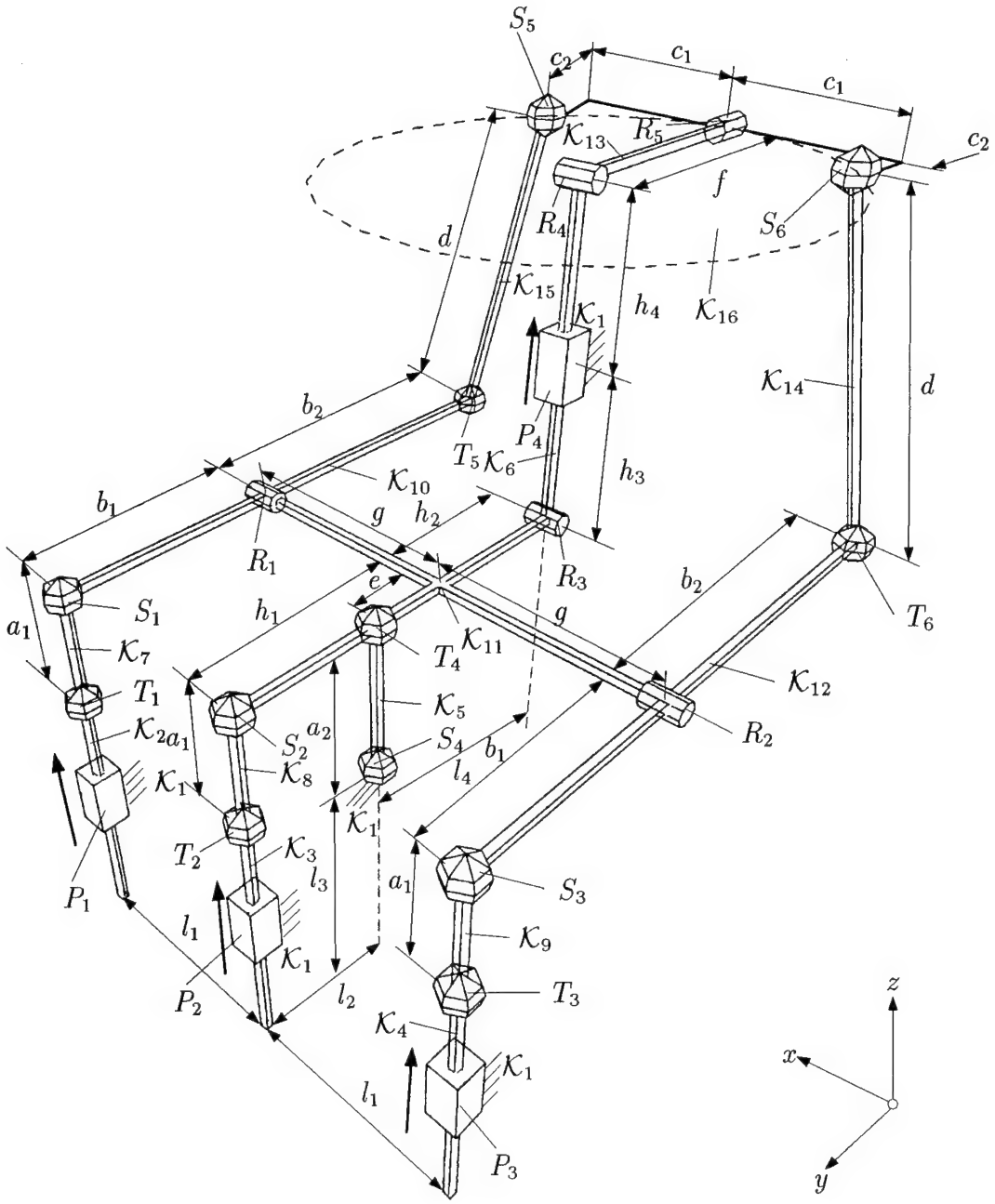



Figure 45: Schematic view of the mixer unit

```
<< GlobalKinematics.m
K          = Table[Null,{16},{16}];
K[[1,2]]   = ETransform[1,1,l1].ETransform[3,1,s[1]];
K[[1,3]]   = ETransform[3,1,s[2]];
K[[1,4]]   = ETransform[1,1,-l1].ETransform[3,1,s[3]];
K[[1,5]]   = ETransform[2,1,-l2].ETransform[3,1,l3].ETransform[3,0,-phi[12]].
             ETransform[2,0,-phi[11]].ETransform[1,0,-phi[10]].
             ETransform[3,1,a2/2];
K[[1,6]]   = ETransform[2,1,-l2-l4].ETransform[3,1,s[4]];
K[[2,7]]   = ETransform[1,0,beta[1]].ETransform[2,0,beta[2]].
             ETransform[3,1,a1/2];
K[[3,8]]   = ETransform[1,0,beta[3]].ETransform[2,0,beta[4]].
             ETransform[3,1,a1/2];
K[[4,9]]   = ETransform[1,0,beta[5]].ETransform[2,0,beta[6]].
             ETransform[3,1,a1/2];
K[[7,10]]  = ETransform[3,1,a1/2].ETransform[1,0,phi[1]].ETransform[2,0,phi[2]].
             ETransform[3,0,phi[3]].ETransform[2,1,-b1];
K[[5,11]]  = ETransform[3,1,a2/2].ETransform[2,0,-beta[8]].
             ETransform[1,0,-beta[7]].ETransform[2,1,-e];
K[[8,11]]  = ETransform[3,1,a1/2].ETransform[1,0,phi[4]].ETransform[2,0,phi[5]].
             ETransform[3,0,phi[6]].ETransform[2,1,-h1];
K[[9,12]]  = ETransform[3,1,a1/2].ETransform[1,0,phi[7]].ETransform[2,0,phi[8]].
             ETransform[3,0,phi[9]].ETransform[2,1,-b1];
K[[11,6]]  = ETransform[2,1,-h2].ETransform[1,0,-delta[3]].ETransform[3,1,h3];
K[[10,15]] = ETransform[2,1,-b2].ETransform[1,0,beta[9]].
             ETransform[2,0,beta[10]].ETransform[3,1,d/2];
K[[12,14]] = ETransform[2,1,-b2].ETransform[1,0,beta[11]].
             ETransform[2,0,beta[12]].ETransform[3,1,d/2];
K[[6,13]]  = ETransform[3,1,h4].ETransform[1,0,delta[4]].ETransform[2,1,-f/2];
K[[13,16]] = ETransform[2,1,-f/2].ETransform[2,0,delta[5]];
K[[16,15]] = ETransform[1,1,c1].ETransform[2,1,c2].ETransform[1,0,phi[13]].
             ETransform[2,0,phi[14]].ETransform[3,0,phi[15]].
             ETransform[3,1,-d/2];
K[[16,14]] = ETransform[1,1,-c1].ETransform[2,1,c2].ETransform[1,0,phi[16]].
             ETransform[2,0,phi[17]].ETransform[3,0,phi[18]].
             ETransform[3,1,-d/2];
K[[10,11]] = ETransform[1,0,-delta[1]].ETransform[1,1,-g];
K[[12,11]] = ETransform[1,0,-delta[2]].ETransform[1,1,g];
Joints     = {s[1],s[2],s[3],s[4],
              beta[1],beta[2],beta[3],beta[4],beta[5],beta[6],
              beta[7],beta[8],beta[9],beta[10],beta[11],beta[12],
              phi[1],phi[2],phi[3],phi[4],phi[5],phi[6],phi[7],phi[8],phi[9],
              phi[10],phi[11],phi[12],phi[13],phi[14],phi[15],phi[16],phi[17],
              phi[18],delta[1],delta[2],delta[3],delta[4],delta[5]}
Bodies     = {2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
SolveGlobalKinematics[K,Joints,{},Bodies]
```

Figure 46: Input data for a Mathematica session of the mixer unit

With the user-supplied frame adjacency matrix, the resulting graph takes on the from depicted in Fig. 47. Clearly, the system consists of just one cluster, or leaf. A suitable minimum cycle basis consists of following six loops

$$\left.\begin{aligned}
L_1 &= \{e_{6,11}, e_{11,10}, e_{10,15}, e_{15,16}, e_{16,13}, e_{13,6}\} \\
L_2 &= \{e_{6,11}, e_{11,12}, e_{12,14}, e_{14,16}, e_{16,13}, e_{13,6}\} \\
L_3 &= \{e_{6,11}, e_{11,8}, e_{8,3}, e_{3,1}, e_{1,6}\} \\
L_4 &= \{e_{6,11}, e_{11,5}, e_{5,1}, e_{1,6}\} \\
L_5 &= \{e_{6,11}, e_{11,10}, e_{10,7}, e_{7,2}, e_{2,1}, e_{1,6}\} \\
L_6 &= \{e_{6,11}, e_{11,12}, e_{12,9}, e_{9,4}, e_{4,1}, e_{1,6}\}
\end{aligned}\right\}$$

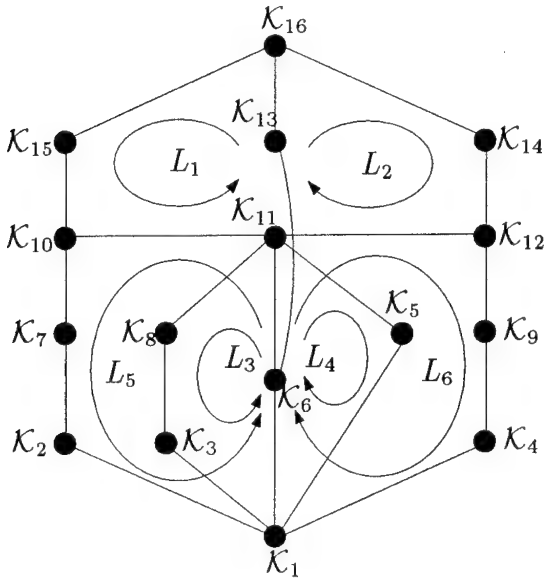



Figure 47: Interconnection graph of the mixer unit

The loops have been drawed into the scheme of the mixing unit in Fig. 48. The corresponding kinematical network with solution flow is shown in Fig. 49. The program determines as inputs for the system the joint variables $\delta_4$, $\delta_3$ and $\beta_{12}$, which do not coincide with the pilot inputs. This is because, by this particular choice, the algorithm can guarantee a closed-form solution for the complete kinematics, whereas by the selection of the pilot inputs, only an iterative solution is possible. The closed-form solution is based on a recursive processing of the loops of the system, starting with the computation of $\delta_2$ and $\delta_5$ in $L_2$, then resolving loops $L_1$ and $L_4$ for $\delta_1$ and $s_4$, and with these evaluating the kinematics of $L_3$, $L_5$ and $L_6$. As each loop is solvable in closed form, the complete process also renders analytical solutions. With this resolution scheme, the complete relative kinematics becomes recursively solvable, and corresponding expressions for velocities and accelerations can be generated.

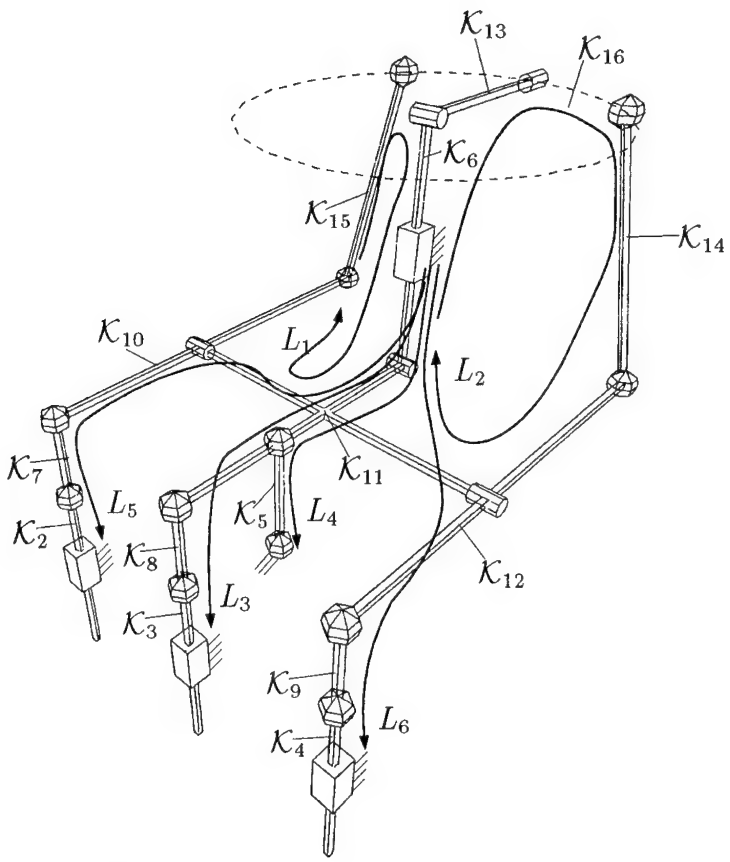



Figure 48: Loops of the mixer unit

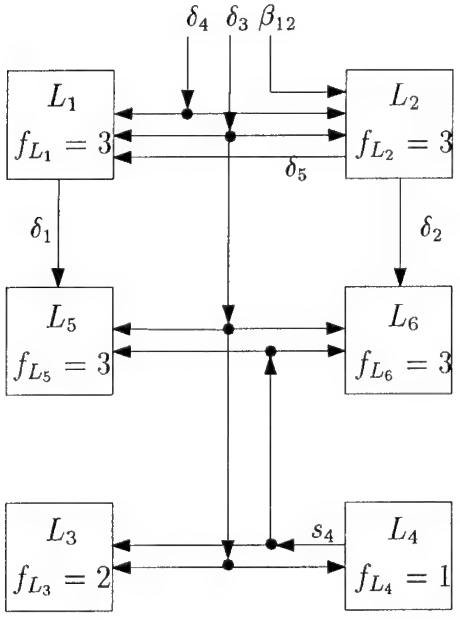



Figure 49: Kinematical network of the mixer unit

For the generation of the absolute kinematics, the topological structure-processing scheme described in Section **10** is carried out. The corresponding system matrices $R_a$, $C_a$, $R_c$ and $C_c$ become:

$$C_a = \begin{array}{c} \begin{matrix} \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 & \mathcal{K}_6 & \mathcal{K}_7 & \mathcal{K}_8 & \mathcal{K}_9 & \mathcal{K}_{11} & \mathcal{K}_{10} & \mathcal{K}_{12} & \mathcal{K}_{13} & \mathcal{K}_{14} & \mathcal{K}_{15} & \mathcal{K}_{16} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} e_{1,2} \\ e_{1,3} \\ e_{1,4} \\ e_{1,5} \\ e_{1,6} \\ e_{2,7} \\ e_{3,8} \\ e_{4,9} \\ e_{6,11} \\ e_{11,10} \\ e_{11,12} \\ e_{6,13} \\ e_{12,14} \\ e_{10,15} \\ e_{13,16} \end{matrix}$$

$$C_c = \begin{array}{c} \begin{matrix} \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 & \mathcal{K}_6 & \mathcal{K}_7 & \mathcal{K}_8 & \mathcal{K}_9 & \mathcal{K}_{11} & \mathcal{K}_{10} & \mathcal{K}_{12} & \mathcal{K}_{13} & \mathcal{K}_{14} & \mathcal{K}_{15} & \mathcal{K}_{16} \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \end{array} \begin{matrix} e_{5,11} \\ e_{7,10} \\ e_{8,11} \\ e_{9,12} \\ e_{16,14} \\ e_{16,15} \end{matrix}$$

$$R_a = \begin{array}{c} \begin{matrix} \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 & \mathcal{K}_6 & \mathcal{K}_7 & \mathcal{K}_8 & \mathcal{K}_9 & \mathcal{K}_{11} & \mathcal{K}_{10} & \mathcal{K}_{12} & \mathcal{K}_{13} & \mathcal{K}_{14} & \mathcal{K}_{15} & \mathcal{K}_{16} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} e_{1,2} \\ e_{1,3} \\ e_{1,4} \\ e_{1,5} \\ e_{1,6} \\ e_{2,7} \\ e_{3,8} \\ e_{4,9} \\ e_{6,11} \\ e_{11,10} \\ e_{11,12} \\ e_{6,13} \\ e_{12,14} \\ e_{10,15} \\ e_{13,16} \end{matrix}$$

| Task | Topology | Position | Velocity | Acceler. | Total |
|---|---|---|---|---|---|
| determination of clusters | 6.84 s | | | | 6.84 s |
| loop detection | 202.15 s | | | | 202.15 s |
| kinematical networks | 12.50 s | | | | 12.50 s |
| loop 1 | | 142.28 s | 63.85 s | 5.77 s | 211.90 s |
| loop 2 | | 140.26 s | 5.06 s | 3.17 s | 148.49 s |
| loop 3 | | 20.91 s | 11.68 s | 3.20 s | 35.79 s |
| loop 4 | | 27.84 s | 9.53 s | 2.75 s | 40.12 s |
| loop 5 | | 85.89 s | 44.29 s | 4.40 s | 134.58 s |
| loop 6 | | 93.69 s | 46.69 s | 4.43 s | 144.81 s |
| equation reorderring | | | | | 416.99 s |
| SOLVAS-matrices | 0.73 s | | | | 0.73 s |
| absolute kinematics | 0.15 s | 63.69 s | 46.56 s | 21.60 s | 132.00 s |
| total | 222.37 | 574.56 | 227.66 | 45.32 | 1486.90 s |

Table 11: CPU-time for generation of symbolical equation for the mixer unit

$$R_c = \begin{array}{c} \begin{matrix} \mathcal{K}_2 & \mathcal{K}_3 & \mathcal{K}_4 & \mathcal{K}_5 & \mathcal{K}_6 & \mathcal{K}_7 & \mathcal{K}_8 & \mathcal{K}_9 & \mathcal{K}_{11} & \mathcal{K}_{10} & \mathcal{K}_{12} & \mathcal{K}_{13} & \mathcal{K}_{14} & \mathcal{K}_{15} & \mathcal{K}_{16} \end{matrix} \\ \left[\begin{array}{cccccccccccccccc} 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & -1 & 1 \end{array}\right] \end{array} \begin{matrix} e_{5,11} \\ e_{7,10} \\ e_{8,11} \\ e_{9,12} \\ e_{16,14} \\ e_{16,15} \end{matrix}$$

With the knowledge of shortest-path matrix $R_a$, the absolute kinematics amounts to superposing the relative motions along the paths according to the equations for serial chains derived in Section **2.3**. With these equations, the kinematical processing is terminated.

The complete kinematical processing described above took about 25 min. on a workstation of type HP 9000/720. Table 11 displays an overview of the distribution of computational time to the individual steps. During this process, a total of 1437 symbolical expressions were generated, which are roughly divided equally between relative and absolute kinematics (Table 12). The number of operations involved in these expressions is reproduced in Table 13.

A complete listing of all expressions, including the user inputs, is provided in Appendix A.

| | |
|---|---|
| relative kinematics | |
| equations for joint coordinates and their derivatives | 108 |
| substitutions | 692 |
| absolute kinematics | |
| equations for the kinematical state of all bodies | 360 |
| substitutions | 277 |
| total | 1437 |

Table 12: Number of symbolical equations for the mixer unit

| Type | Number |
|---|---|
| Addition and subtraction | 3551 |
| Multiplication and division | 4879 |
| Power | 61 |
| Square root | 5 |
| Trigonometric functions | 33 |
| Inverse trigonometric functions | 29 |
| Total | 8558 |

Table 13: Total number of operations for the mixer unit

# 12 Conclusions

The methods described in this report show that it is feasible to design automated procedures that produce closed-form solutions of the kinematics for multiple-loop mechanisms at position, velocity and acceleration level. The generation of such solutions is made possible by analysing the topology and the geometry of the system at hand, performing a successive dissection of the system, first in clusters of recursively solvable and non-recursively solvable subsystems, then in sets of independent loops, and finally in sequences of transformations which leave certain geometric elements invariant. By representing the global system of equations as a concatenation of such simpler subsystems, the subsequent re-assembly of the local solutions poses no problem, and an integrated program can be written that carries out all of these steps in a completely autonomous manner. The report describes such a program, called SYMKIN, showing also that the aforementioned optimized solutions can in fact be achieved in acceptable processing time for a number of rather complex spatial systems. Such optimized solutions may serve as a practicable means to design programs suited for real-time simulation of complex vehicle dynamics in the near future.

# 13 List of Project-Related Publications and Interim Reports

During the period of the present project, the following related publications appeared:

- **Th. Krupp, A. Kecskemethy and R. Wehage.** (1994) Precomputed Loop-based Kinematics and Dynamics in Large Scale Vehicle Models. In Halin, J., Karplus, W. and Rimane,R. (eds.) *Proceedings of the First Joint Conference of International Simulation Societies*, CISS, Zürich.
- 1st Interim Report: Decomposition of Multiloop Mechanisms Into Recursively and Non-Recursively Solvable Subsystems. (27.9.1994).
- 2nd Interim Report: Automatic Generation of Closed-Form Solutions of Multiple-Loop Mechanisms and Symbolic Velocity Processing for a Single Loop. (23.12.94).
- 3rd Interim Report: Automatic Generation of Closed-Form Solutions and Topological Analysis of Multiple-Loop Mechanisms. (28.03.95).

# 14 List of Participating Scientific Personnel

The following scientific personnel participated in the present project:

- Dr.-Ing. Andrés Kecskeméthy, project supervisor. No advanced degrees earned while employed on the project.
- Dipl.-Ing. Thorsten Krupp, research assistance. No advanced degrees earned while employed on the project.

# References


**Angeles, J.** (1988) *Rational Kinematics.* Springer Tracts in Natural Philosophy 34. Springer-Verlag, New York.

**Andrews, G. C. and Kesavan, H. K.** (1975). The vector network model: A new approach to vector dynamics. *Mechanism and Machine Theory*, 10:509–519.

**Brandl, H. Johanni, R. and Otter, M.** (1986). A very efficient algorithm for the aimulation of robots and similar multibody systems without inversion of the mass matrix. In *Proceedings of the IFAC/IFIP/IMACS Symposium on Theory of Robotics Research*, Vienna, Austria.

**Carré, B.** (1979). *Graphs and Networks.* Oxford University Press.

**Denavit, J. and Hartenberg, R.** (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Transactions of the ASME, Journal of Applied Mechanics*, 215–221.

**Fanghella, P.** (1988). Kinematics of spatial linkages by group algebra: A structure-based approach. *Mechanism and Machine Theory*, 23(3):171–183.

**Fanghella, P. and Galletti, C.** (1993). A modular approach for computational kinematics. In Angeles, J., Hommel, G. and Kovacs, P. (editors), *Computational Kinematics, Solid Mechanics and its Applications*, volume 28. Kluwer Academic Publishers.

**Garcia de Jalón, J., Avello, A. and Jiménez, J.** (1987). Basis for the nullspace for the Jacobian matrix of constrained multibody systems. In Bautista, E., Garcia-Lomas, J. and Navarro, A. (editor), *Proceedings 7th World Congress Th. Mach. Mech.*, pages 501–504, Sevilla. Pergamon Press.

**Gondran, M. and Minoux, M.** (1984). *Graphs and Algorithms – Wiley-Interscience Series in Discrete Mathematics.* John Wiley & Sons.

**Halmos, P. R.** (1987) *Finite-Dimensional Vector Spaces.* Undergraduate Texts in Mathematics. Springer-Verlag, New York, Heidelberg, Berlin.

**Hiller, M. and Anantharaman, M.** (1989) Systematische Strukturierung der Bindungsgleichungen mehrschleifiger Mechanismen. *Zeitschrift für angewandte Mathematik und Mechanik,* 69:T303–T305.

**Hiller, M. and Kecskeméthy, A.** (1989) Equations of motion of complex multibody systems using kinematical differentials. *Transactions of the Canadian Society of Mechanical Engineers*, 13(4):113–121.

**Hiller, M., Kecskeméthy, A. and Woernle, C.** (1985). A loop-based kinematical analysis of spatial mechanisms. ASME-Paper No. 86-DET-184.

**Horton, J.** (1987) A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal of Computing*, 16(2):358–366.

**Hubicka, E. and Syslo, M. M.** (1975) Minimal bases of cycles of a graph. In *Proc. 2nd Czechoslovak Symposium on Graph Theory*, 283–293, Prag. Academia.

**Hunt, K. H.** (1986) The particular of the general? (Some examples from robot kinematics). *Mechanism and Machine Theory.* 21(6):481–487.

**Jiménez, J., Avello, A., de Jalón, J. G. and Avello, A.** (1993). An efficient implementation of the velocity transformation method for real-time dynamics with illustrative examples. In Pereira, M. and Ambrósio, J. (editor), *Proceedings of the NATO-Advanced Study Institute on Computer Aided Analysis of Rigid and Flexible Mechanical Systems, Volume II (Contributed Papers)*, 39–58, Tróia, Portugal.

**Kane, T.R. and Levinson, D.A.** (1985). *Dynamics: Theory and Applications.* McGraw-Hill Series in Mechanical Engineering. McGraw-Hill Book Company, New York.

**Kecskeméthy, A.** (1993a). *Objektorientierte Modellierung der Dynamik von Mehrkörpersystemen mit Hilfe von Übertragungselementen.* Fortschrittberichte VDI Reihe 20 Nr. 88. VDI Verlag, Düsseldorf.

**Kecskeméthy, A.** (1993b). On closed form solutions of multiple-loop mechanisms. In Angeles, J., Hommel, G., and Kovacs, P. (editors) *Computational Kinematics.* Kluwer Academic Publishers.

**Kecskeméthy, A. and Hiller, M.** (1992). Automatic closed-form kinematics-solutions for recursive single-loop chains. In *Flexible Mechanisms, Dynamics, and Analysis, Proc. of the 22nd Biennal ASME-Mechanisms Conference, Scottsdale (USA)*, 387–393.

**Kreuzer, E.** (1979). Symbolische Berechnung der Bewegungsgleichungen von Mehrkörpersystemen. *Fortschrittberichte VDI Reihe 11 Nr. 32*, VDI-Verlag, Düsseldorf.

**Krupp, Th.** (1992). Objektorientierte Erstellung der Jacobimatrizen von Mehrkörpersystemen durch Verkettung dünnbesetzter Teilabbildungen – Einsatz bei der Generierung der Bewegungsgleichungen. Diploma Thesis, Gerhard-Mercator-Universität – GH – Duisburg.

**Lee, H., Woernle, C. and Hiller, M.** (1990). A complete solution for the inverse kinematic problem of the general 6R robot manipulator. In *Proceedings of the ASME-Conference on Robotics, Cincinnati.*

**Nielan, P. E. and Kane, T. R.** (1986). Symbolic generation of efficient simulation/control routines for multibody systems. In Bianchi, G. and Schiehlen, W. (editor), *Dynamics of Multibody Systems. IUTAM/IFToMM Symposium, Udine (Italy) 1985*, pages 153–164. Springer-Verlag, Berlin.

**Nikravesh, P. and Haug, E.** (1982). Generalized coordinate partitioning for analysis of mechanical systems with nonholonomic constraints. ASME-Paper No. 82-DET-94.

**Nikravesh, P.** (1988) *Computer-Aided Analysis of Mechanical Systems.* Prentice Hall.

**Olver, P. J.** (1986). *Applications of Lie Groups to Differential Equations.* Graduate Texts in Mathematics 107. Springer-Verlag.

**Orin, D. and Schrader, W.** (1983). Efficient jacobian determination for robot manipulators. In *Proceedings of the Sith World Congress on Theory of Machines and Mechanisms*, 994–997.

**Orlandea, N.** (1987). ADAMS (Theory and applications). In DePater, A. and Pacejka, H. (editor), *Proc. 3rd Seminar on Advanced Vehicle Systems Dynamics (Amalfi, Mai 1986)*, 121–166.

**Orlandea, N., Chase, M. and Calahan, D.** (1979). A sparsity-oriented approach to the dynamic analysis and design of mechanical design of mechanical systems – parts 1 & 2. *Transactions of the ASME, Journal of Engineering for Industry*, 773–784.

**Raghavan, M. and Roth, B.** (1993). Inverse kinematics of the general 6R manipulator and related linkages. *Transactions of the ASME, Journal of Mechanical Design*, 115(3).

**Renaud, M.** (1981). Geometric and kinematic models of a robot manipulator: calculation of the Jacobian. In *11th ISIR, Tokyo, Japan*, 757–763.

**Schneider, M. and Hiller, M.** (1995) Modellbildung und Regelung eines elastischen, hydraulisch angetriebenen Großmanipulators, ZAMM 75, SI-S127-S128.

**Tsai, F. and Haug, E.** (1991a). Real-time multibody system dynamic simulation, Part I—A modified recursive formulation and topological analysis. *Mechanics, Structures and Machines*, 19(1).

**Tsai, F. and Haug, E.** (1991b). Real-time multibody system dynamic simulation, Part I—A parallel algorithm and numerical results. *Mechanics, Structures and Machines*, 19(2).

**Waldron, K.J.** (1981) Geometrically based manipulator rate control algorithms. The Ohio State University.

**Walker, M. and Orin, D.** (1982). Efficient dynamic computer simulation of robotic mechanisms. *J. Dyn. Syst. Meas. Contr.*, 104:205–211.

**Wehage, R. and Belczynski, M.** (1992). High resolution vehicle simulations using precomputed coefficients. In Rizzoni, G., El-Gindy, M., Wong, J. and Zeid, A. (editors), *Transportation Systems-ASME, DCS*, volume 44, pages 311–325.

**Wehage, R. and Belczynski, M.** (1993). Constrained multibody dynamics. In Pereira, M. and Ambrósio, J. (editors), *Proceedings of the NATO-Advanced Study Institute on Computer Aided Analysis of Rigid and Flexible Mechanical Systems, Volume I (Main Lectures)*, 461–487, Tróia, Portugal.

**Wehage, R. and Haug, E.** (1982). Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic sysems. *Transactions of the ASME, Journal of Mechanical Design*, No. 104, 247–255.

**Wehage, R. A. and Janosi, Z. J.** (1993) *Workshop on: Computational Methods in Multibody Vehicle Dynamics.* Presented at: Godollo University of Agriculture, May 17 to May 18, 1993.

**Wittenburg, J.** (1977). *Dynamics of Systems of Rigid Bodies*, volume 33 of Leitfaden der angewandten Mathematik and Mechanik. B. G. Teubner, Stuttgart.

**Woernle, C.** (1988). *Ein systematisches Verfahren zur Aufstellung der geometrischen Schließbedingungen in kinematischen Schleifen mit Anwendung bei der Rückwärtstransformation für Industrieroboter.* Fortschrittberichte VDI Reihe 18 Nr. 59. VDI Verlag, Düsseldorf.

**Wolfram S.** (1988). *Mathematica. A System for Doing Mathematics by Computer.* Addison-Wesley Publishing Company.

# A Closed Form Solutions for the Mixer Unit

```
ADH7[1, 4] -> d Sin[beta[12]]
                                    2
                                   c1
delta[5] -> ArcTan[Sqrt[-1 + ------------------]] + ArcTan[c1, 0]
                                    2
                             (g - ADH7[1, 4])
ADH5[3, 4] -> -(f Sin[delta[4]])
ADH4[3, 4] -> h3 + h4 + ADH5[3, 4]
ADH5[2, 4] -> -(f Cos[delta[4]])
ADH3[2, 4] -> ADH5[2, 4] Cos[delta[3]] + ADH4[3, 4] Sin[delta[3]]
ADH11[1, 4] -> -(c1 Cos[delta[5]])
ADH11[3, 4] -> c1 Sin[delta[5]]
ADH2[2, 4] -> -h2 + ADH3[2, 4]
ADH3[2, 2] -> Cos[delta[3]] Cos[delta[4]] + Sin[delta[3]] Sin[delta[4]]
ADH3[2, 3] -> Cos[delta[4]] Sin[delta[3]] - Cos[delta[3]] Sin[delta[4]]
ADH3[3, 2] -> -(Cos[delta[4]] Sin[delta[3]]) + Cos[delta[3]] Sin[delta[4]]
ADH3[3, 3] -> Cos[delta[3]] Cos[delta[4]] + Sin[delta[3]] Sin[delta[4]]
ADH3[3, 4] -> ADH4[3, 4] Cos[delta[3]] - ADH5[2, 4] Sin[delta[3]]
ADH10[1, 4] -> g + ADH11[1, 4]
ADH10[2, 4] -> ADH12[2, 4] + c2 ADH3[2, 2] + ADH11[3, 4] ADH3[2, 3]
ADH10[3, 4] -> c2 ADH3[3, 2] + ADH11[3, 4] ADH3[3, 3] + ADH3[3, 4]
                          2                 2                 2
ADH12[1, 1] -> ADH10[1, 4]  + ADH10[2, 4]  + ADH10[3, 4]
ADH7[3, 4] -> d Cos[beta[12]]
                 2
ADH8[1, 1] -> b2
                        2             2
ADH9[1, 1] -> ADH7[1, 4]  + ADH7[3, 4]
                    -ADH12[1, 1] + ADH8[1, 1] + ADH9[1, 1]
beta[11] -> -ArcSin[---------------------------------------]
                                2 b2 ADH7[3, 4]
ADH14[2, 4] -> -(ADH7[3, 4] Sin[beta[11]])
ADH13[2, 4] -> -b2 + ADH14[2, 4]
ADH14[3, 4] -> ADH7[3, 4] Cos[beta[11]]
delta[2] -> -ArcTan[(ADH10[2, 4] ADH13[2, 4] + ADH10[3, 4] ADH14[3, 4]) /
                    2             2
>      (ADH10[2, 4]  + ADH10[3, 4] ),
>     (-(ADH10[3, 4] ADH13[2, 4]) + ADH10[2, 4] ADH14[3, 4]) /
                    2             2
>      (ADH10[2, 4]  + ADH10[3, 4] )]
ADH18[3, 2] ->
>   -(Cos[delta[2]] Sin[beta[11]]) - Cos[beta[11]] Sin[delta[2]]
ADH18[3, 3] -> Cos[beta[11]] Cos[delta[2]] - Sin[beta[11]] Sin[delta[2]]
ADH17[1, 2] -> -(ADH18[3, 2] Sin[beta[12]])
ADH17[1, 3] -> -(ADH18[3, 3] Sin[beta[12]])
ADH18[2, 2] -> Cos[beta[11]] Cos[delta[2]] - Sin[beta[11]] Sin[delta[2]]
ADH18[2, 3] -> Cos[delta[2]] Sin[beta[11]] + Cos[beta[11]] Sin[delta[2]]
ADH15[1, 1] ->
>   Cos[beta[12]] Cos[delta[5]] -
>    (ADH17[1, 2] ADH3[2, 3] + ADH17[1, 3] ADH3[3, 3]) Sin[delta[5]]
ADH15[2, 1] ->
>   -((ADH18[2, 2] ADH3[2, 3] + ADH18[2, 3] ADH3[3, 3]) Sin[delta[5]])
                              ADH15[2, 1]
phi[18] -> -3.14159 - ArcTan[-----------]
                              ADH15[1, 1]
ADH17[3, 2] -> ADH18[3, 2] Cos[beta[12]]
ADH17[3, 3] -> ADH18[3, 3] Cos[beta[12]]
ADH15[3, 1] ->
>   Cos[delta[5]] Sin[beta[12]] -
>    (ADH17[3, 2] ADH3[2, 3] + ADH17[3, 3] ADH3[3, 3]) Sin[delta[5]]
ADH21[1, 1] -> ADH15[1, 1] Cos[phi[18]] - ADH15[2, 1] Sin[phi[18]]
                             ADH21[1, 1]
phi[17] -> ArcTan[----------------------------,
                                2             2
                   ADH15[3, 1]  + ADH21[1, 1]
             ADH15[3, 1]
>    ---------------------------]
                2             2
     ADH15[3, 1]  + ADH21[1, 1]
ADH15[1, 2] -> ADH17[1, 2] ADH3[2, 2] + ADH17[1, 3] ADH3[3, 2]
ADH15[2, 2] -> ADH18[2, 2] ADH3[2, 2] + ADH18[2, 3] ADH3[3, 2]
ADH15[3, 2] -> ADH17[3, 2] ADH3[2, 2] + ADH17[3, 3] ADH3[3, 2]
ADH23[3, 1] -> -(Cos[phi[18]] Sin[phi[17]])
ADH23[3, 2] -> Sin[phi[17]] Sin[phi[18]]
ADH22[2, 2] -> ADH15[2, 2] Cos[phi[18]] + ADH15[1, 2] Sin[phi[18]]
ADH22[3, 2] ->
>   ADH15[1, 2] ADH23[3, 1] + ADH15[2, 2] ADH23[3, 2] +
>    ADH15[3, 2] Cos[phi[17]]
phi[16] -> -ArcTan[ADH22[2, 2], ADH22[3, 2]]
ADH37[3, 4] -> -(b2 Sin[beta[11]])
ADH36[3, 4] -> ADH37[3, 4] Cos[beta[12]]
ADH29[2, 4] -> (-h3 - h4) Sin[delta[4]]
ADH29[3, 4] -> (-h3 - h4) Cos[delta[4]]
ADH35[3, 4] -> -d + ADH36[3, 4]
ADH36[1, 4] -> -(ADH37[3, 4] Sin[beta[12]])
ADH37[2, 4] -> b2 Cos[beta[11]]
ADH23[1, 1] -> Cos[phi[17]] Cos[phi[18]]
ADH25[2, 4] -> -c2 + f
ADH28[1, 4] -> c1 - ADH29[3, 4] Sin[delta[5]]
ADH28[2, 4] -> -c2 + f + ADH29[2, 4]
ADH28[3, 4] -> ADH29[3, 4] Cos[delta[5]]
ADH31[1, 4] -> -(d Sin[phi[17]])
ADH31[2, 4] -> d Cos[phi[17]] Sin[phi[16]]
ADH31[3, 4] -> -(d Cos[phi[16]] Cos[phi[17]])
ADH32[1, 4] -> -(d Sin[phi[17]])
ADH32[2, 4] -> d Cos[phi[17]] Sin[phi[16]]
ADH32[3, 4] -> -(d Cos[phi[16]] Cos[phi[17]])
ADH34[2, 4] ->
>   -(ADH35[3, 4] Cos[phi[17]] Sin[phi[16]]) +
>    ADH37[2, 4] (Cos[phi[16]] Cos[phi[18]] - ADH23[3, 2] Sin[phi[16]]) +
>    ADH36[1, 4] (-(ADH23[3, 1] Sin[phi[16]]) + Cos[phi[16]] Sin[phi[18]])
ADH34[3, 4] ->
>   ADH35[3, 4] Cos[phi[16]] Cos[phi[17]] +
>    ADH37[2, 4] (ADH23[3, 2] Cos[phi[16]] + Cos[phi[18]] Sin[phi[16]]) +
>    ADH36[1, 4] (ADH23[3, 1] Cos[phi[16]] + Sin[phi[16]] Sin[phi[18]])
ADH23[1, 2] -> -(Cos[phi[17]] Sin[phi[18]])
ADH34[1, 4] ->
>   ADH23[1, 1] ADH36[1, 4] + ADH23[1, 2] ADH37[2, 4] +
>    ADH35[3, 4] Sin[phi[17]]
ADH30[2, 3] -> -(Cos[phi[17]] Sin[phi[16]])
ADH30[3, 3] -> Cos[phi[16]] Cos[phi[17]]
ADH31[2, 2] -> Cos[phi[16]] Cos[phi[18]] - ADH23[3, 2] Sin[phi[16]]
ADH31[3, 2] -> ADH23[3, 2] Cos[phi[16]] + Cos[phi[18]] Sin[phi[16]]
ADH32[1, 1] -> ADH23[1, 1] Cos[beta[12]] + Sin[beta[12]] Sin[phi[17]]
ADH32[2, 1] ->
>   -(Cos[phi[17]] Sin[beta[12]] Sin[phi[16]]) +
>    Cos[beta[12]] (-(ADH23[3, 1] Sin[phi[16]]) +
>       Cos[phi[16]] Sin[phi[18]])
ADH32[3, 1] ->
>   Cos[phi[16]] Cos[phi[17]] Sin[beta[12]] +
>   Cos[beta[12]] (ADH23[3, 1] Cos[phi[16]] + Sin[phi[16]] Sin[phi[18]])
ADH34[1, 1] -> ADH23[1, 1] Cos[beta[12]] + Sin[beta[12]] Sin[phi[17]]
ADH34[2, 1] ->
>   -(Cos[phi[17]] Sin[beta[12]] Sin[phi[16]]) +
>    Cos[beta[12]] (-(ADH23[3, 1] Sin[phi[16]])) +
>       Cos[phi[16]] Sin[phi[18]])
ADH34[3, 1] ->
>   Cos[phi[16]] Cos[phi[17]] Sin[beta[12]] +
>    Cos[beta[12]] (ADH23[3, 1] Cos[phi[16]] + Sin[phi[16]] Sin[phi[18]])
JACR2[4] -> ADH25[2, 4] Sin[delta[5]]
JACR2[5] -> -(c1 Sin[delta[5]])
JACR2[6] -> -(ADH25[2, 4] Cos[delta[5]])
JACR3[4] -> ADH28[2, 4] Sin[delta[5]]
JACR3[5] -> ADH28[3, 4] Cos[delta[5]] - ADH28[1, 4] Sin[delta[5]]
JACR3[6] -> -(ADH28[2, 4] Cos[delta[5]])
JACR7[4] -> ADH31[2, 4] ADH31[3, 2] - ADH31[2, 2] ADH31[3, 4]
```

```
JACR7[5] -> -(ADH31[1, 4] ADH31[3, 2]) + ADH23[1, 2] ADH31[3, 4]
JACR7[6] -> ADH31[1, 4] ADH31[2, 2] - ADH23[1, 2] ADH31[2, 4]
JACR8[4] -> ADH32[2, 4] ADH32[3, 1] - ADH32[2, 1] ADH32[3, 4]
JACR8[5] -> -(ADH32[1, 4] ADH32[3, 1]) + ADH32[1, 1] ADH32[3, 4]
JACR8[6] -> ADH32[1, 4] ADH32[2, 1] - ADH32[1, 1] ADH32[2, 4]
JACR9[4] -> ADH34[2, 4] ADH34[3, 1] - ADH34[2, 1] ADH34[3, 4]
JACR9[5] -> -(ADH34[1, 4] ADH34[3, 1]) + ADH34[1, 1] ADH34[3, 4]
JACR9[6] -> ADH34[1, 4] ADH34[2, 1] - ADH34[1, 1] ADH34[2, 4]
(beta[11])' ->
>   (-(JACR7[5] JACR9[4] (beta[12]))') + JACR7[4] JACR9[5] (beta[12])' -
>      JACR3[5] JACR9[4] (delta[3])' + JACR3[4] JACR9[5] (delta[3])' +
>      JACR2[5] JACR9[4] (delta[4])' - JACR2[4] JACR9[5] (delta[4])') /
>    (JACR8[5] JACR9[4] - JACR8[4] JACR9[5])
(delta[2])' ->
>   -((JACR8[4] (beta[11])' + JACR7[4] (beta[12])' +
>       JACR3[4] (delta[3])' - JACR2[4] (delta[4])') / JACR9[4])
(delta[5])' ->
>   (JACR8[6] (beta[11])' + JACR7[6] (beta[12])' + JACR9[6] (delta[2])' +
>      JACR3[6] (delta[3])' - JACR2[6] (delta[4])') / ci
(phi[17])' -> (ADH30[3, 3] ADH32[2, 1] (beta[11])' -
>      ADH30[2, 3] ADH32[3, 1] (beta[11])' +
>      ADH30[3, 3] ADH31[2, 2] (beta[12])' -
>      ADH30[2, 3] ADH31[3, 2] (beta[12])' +
>      ADH30[3, 3] ADH34[2, 1] (delta[2])' -
>      ADH30[2, 3] ADH34[3, 1] (delta[2])' -
>      ADH30[2, 3] Sin[delta[5]] (delta[3])' +
>      ADH30[2, 3] Sin[delta[5]] (delta[4])' - ADH30[3, 3] (delta[5])') /
>    (ADH30[3, 3] Cos[phi[16]] - ADH30[2, 3] Sin[phi[16]])
(phi[18])' -> (ADH32[2, 1] (beta[11])' + ADH31[2, 2] (beta[12])' +
>      ADH34[2, 1] (delta[2])' - (delta[5])' - Cos[phi[16]] (phi[17])') /
>    ADH30[2, 3]
(phi[16])' -> ADH32[1, 1] (beta[11])' + ADH23[1, 2] (beta[12])' +
>    ADH34[1, 1] (delta[2])' + Cos[delta[5]] (delta[3])' -
>    Cos[delta[5]] (delta[4])' - Sin[phi[17]] (phi[18])'
JACR10[4] -> JACR3[6] (delta[3])' - JACR2[6] (delta[4])' - ci (delta[5])'
JACR10[6] -> -(JACR3[4] (delta[3])') + JACR2[4] (delta[4])'
JACR11[1] -> -(Sin[delta[5]] (delta[5])')
JACR11[3] -> Cos[delta[5]] (delta[5])'
JACR11[4] -> -(JACR3[5] Sin[delta[5]] (delta[3])') +
>    JACR2[5] Sin[delta[5]] (delta[4])' - JACR2[6] (delta[5])'
JACR11[5] -> (-(Cos[delta[5]] JACR3[6]) + JACR3[4] Sin[delta[5]])
>     (delta[3])' - (-(Cos[delta[5]] JACR2[6]) + JACR2[4] Sin[delta[5]])
>     (delta[4])'
JACR11[6] -> Cos[delta[5]] JACR3[5] (delta[3])' -
>    Cos[delta[5]] JACR2[5] (delta[4])' + JACR2[4] (delta[5])'
JACR12[1] -> -(Sin[delta[5]] (delta[5])')
JACR12[3] -> Cos[delta[5]] (delta[5])'
JACR12[4] -> -(JACR3[5] Sin[delta[5]] (delta[3])') +
>    JACR3[5] Sin[delta[5]] (delta[4])' - JACR3[6] (delta[5])'
JACR12[5] -> (-(Cos[delta[5]] JACR3[6]) + JACR3[4] Sin[delta[5]])
>     (delta[3])' - (-(Cos[delta[5]] JACR3[6]) + JACR3[4] Sin[delta[5]])
>     (delta[4])'
JACR12[6] -> Cos[delta[5]] JACR3[5] (delta[3])' -
>    Cos[delta[5]] JACR3[5] (delta[4])' + JACR3[4] (delta[5])'
JACR14[5] -> JACR8[6] (beta[11])' + JACR7[6] (beta[12])' +
>    JACR9[6] (delta[2])'
JACR14[6] -> -(JACR8[5] (beta[11])') - JACR7[5] (beta[12])' -
>    JACR9[5] (delta[2])'
JACR15[2] -> -(Sin[phi[16]] (phi[16])')
JACR15[3] -> Cos[phi[16]] (phi[16])'
JACR15[4] -> -((Cos[phi[16]] JACR8[6] - JACR8[5] Sin[phi[16]])
>       (beta[11])') - (Cos[phi[16]] JACR7[6] - JACR7[5] Sin[phi[16]])
>     (beta[12])' - (Cos[phi[16]] JACR9[6] - JACR9[5] Sin[phi[16]])
>     (delta[2])'
JACR15[5] -> -(JACR8[4] Sin[phi[16]] (beta[11])') -
>    JACR7[4] Sin[phi[16]] (beta[12])' - JACR9[4] Sin[phi[16]] (delta[2])'
JACR15[6] -> Cos[phi[16]] JACR8[4] (beta[11])' +
>    Cos[phi[16]] JACR7[4] (beta[12])' + Cos[phi[16]] JACR9[4] (delta[2])'
JACR16[1] -> (ADH30[3, 3] Cos[phi[16]] - ADH30[2, 3] Sin[phi[16]])
>    (phi[17])'
JACR16[2] -> -(ADH30[3, 3] (phi[16])') +
>    Sin[phi[16]] Sin[phi[17]] (phi[17])'
JACR16[3] -> ADH30[2, 3] (phi[16])' - Cos[phi[16]] Sin[phi[17]] (phi[17])'
JACR16[4] -> -((-(ADH30[3, 3] JACR8[5]) + ADH30[2, 3] JACR8[6])
>       (beta[11])') - (-(ADH30[3, 3] JACR7[5]) + ADH30[2, 3] JACR7[6])
>     (beta[12])' - (-(ADH30[3, 3] JACR9[5]) + ADH30[2, 3] JACR9[6])
>     (delta[2])'
JACR16[5] -> -((ADH30[3, 3] JACR8[4] - JACR8[6] Sin[phi[17]])
>       (beta[11])') - (ADH30[3, 3] JACR7[4] - JACR7[6] Sin[phi[17]])
>     (beta[12])' - (ADH30[3, 3] JACR9[4] - JACR9[6] Sin[phi[17]])
>     (delta[2])'
JACR16[6] -> -((-(ADH30[2, 3] JACR8[4]) + JACR8[5] Sin[phi[17]])
>       (beta[11])') - (-(ADH30[2, 3] JACR7[4]) + JACR7[5] Sin[phi[17]])
>     (beta[12])' - (-(ADH30[2, 3] JACR9[4]) + JACR9[5] Sin[phi[17]])
>     (delta[2])'
JACR17[1] -> (ADH31[3, 2] Cos[phi[16]] - ADH31[2, 2] Sin[phi[16]])
>     (phi[17])' + (-(ADH30[3, 3] ADH31[2, 2]) + ADH30[2, 3] ADH31[3, 2])
>     (phi[18])'
JACR17[2] -> -(ADH31[3, 2] (phi[16])') +
>    ADH23[1, 2] Sin[phi[16]] (phi[17])' +
>    (ADH23[1, 2] ADH30[3, 3] - ADH31[3, 2] Sin[phi[17]]) (phi[18])'
JACR17[3] -> ADH31[2, 2] (phi[16])' -
>    ADH23[1, 2] Cos[phi[16]] (phi[17])' +
>    (-(ADH23[1, 2] ADH30[2, 3]) + ADH31[2, 2] Sin[phi[17]]) (phi[18])'
JACR17[4] -> -((-(ADH31[3, 2] JACR8[5]) + ADH31[2, 2] JACR8[6])
>       (beta[11])') - (-(ADH31[3, 2] JACR7[5]) + ADH31[2, 2] JACR7[6])
>     (beta[12])' - (-(ADH31[3, 2] JACR9[5]) + ADH31[2, 2] JACR9[6])
>     (delta[2])' + (Cos[phi[16]] JACR7[6] - JACR7[5] Sin[phi[16]])
>     (phi[17])' + (-(ADH30[3, 3] JACR7[5]) + ADH30[2, 3] JACR7[6])
>     (phi[18])'
JACR17[5] -> -((ADH31[3, 2] JACR8[4] - ADH23[1, 2] JACR8[6])
>       (beta[11])') - (ADH31[3, 2] JACR7[4] - ADH23[1, 2] JACR7[6])
>     (beta[12])' - (ADH31[3, 2] JACR9[4] - ADH23[1, 2] JACR9[6])
>     (delta[2])' - JACR7[6] (phi[16])' +
>    JACR7[4] Sin[phi[16]] (phi[17])' +
>    (ADH30[3, 3] JACR7[4] - JACR7[6] Sin[phi[17]]) (phi[18])'
JACR17[6] -> -((-(ADH31[2, 2] JACR8[4]) + ADH23[1, 2] JACR8[5])
>       (beta[11])') - (-(ADH31[2, 2] JACR7[4]) + ADH23[1, 2] JACR7[5])
>     (beta[12])' - (-(ADH31[2, 2] JACR9[4]) + ADH23[1, 2] JACR9[5])
>     (delta[2])' + JACR7[5] (phi[16])' -
>    Cos[phi[16]] JACR7[4] (phi[17])' +
>    (-(ADH30[2, 3] JACR7[4]) + JACR7[5] Sin[phi[17]]) (phi[18])'
JACR18[1] -> -((-(ADH31[3, 2] ADH32[2, 1]) + ADH31[2, 2] ADH32[3, 1])
>       (beta[12])') + (ADH32[3, 1] Cos[phi[16]] -
>       ADH32[2, 1] Sin[phi[16]]) (phi[17])' +
>    (-(ADH30[3, 3] ADH32[2, 1]) + ADH30[2, 3] ADH32[3, 1]) (phi[18])'
JACR18[2] -> -((ADH31[3, 2] ADH32[1, 1] - ADH23[1, 2] ADH32[3, 1])
>       (beta[12])') - ADH32[3, 1] (phi[16])' +
>    ADH32[1, 1] Sin[phi[16]] (phi[17])' +
>    (ADH30[3, 3] ADH32[1, 1] - ADH32[3, 1] Sin[phi[17]]) (phi[18])'
JACR18[3] -> -((-(ADH31[2, 2] ADH32[1, 1]) + ADH23[1, 2] ADH32[2, 1])
>       (beta[12])') + ADH32[2, 1] (phi[16])' -
>    ADH32[1, 1] Cos[phi[16]] (phi[17])' +
>    (-(ADH30[2, 3] ADH32[1, 1]) + ADH32[2, 1] Sin[phi[17]]) (phi[18])'
JACR18[4] -> -((-(ADH32[3, 1] JACR8[5]) + ADH32[2, 1] JACR8[6])
>       (beta[11])') - (-(ADH31[3, 2] JACR8[5]) + ADH31[2, 2] JACR8[6])
>     (beta[12])' - (-(ADH32[3, 1] JACR9[5]) + ADH32[2, 1] JACR9[6])
>     (delta[2])' + (Cos[phi[16]] JACR8[6] - JACR8[5] Sin[phi[16]])
>     (phi[17])' + (-(ADH30[3, 3] JACR8[5]) + ADH30[2, 3] JACR8[6])
>     (phi[18])'
JACR18[5] -> -((ADH32[3, 1] JACR8[4] - ADH32[1, 1] JACR8[6])
>       (beta[11])') - (ADH31[3, 2] JACR8[4] - ADH23[1, 2] JACR8[6])
>     (beta[12])' - (ADH32[3, 1] JACR9[4] - ADH32[1, 1] JACR9[6])
>     (delta[2])' - JACR8[6] (phi[16])' +
>    JACR8[4] Sin[phi[16]] (phi[17])' +
>    (ADH30[3, 3] JACR8[4] - JACR8[6] Sin[phi[17]]) (phi[18])'
JACR18[6] -> -((-(ADH32[2, 1] JACR8[4]) + ADH32[1, 1] JACR8[5])
>       (beta[11])') - (-(ADH31[2, 2] JACR8[4]) + ADH23[1, 2] JACR8[5])
>     (beta[12])' - (-(ADH32[2, 1] JACR9[4]) + ADH32[1, 1] JACR9[5])
>     (delta[2])' + JACR8[5] (phi[16])' -
>    Cos[phi[16]] JACR8[4] (phi[17])' +
>    (-(ADH30[2, 3] JACR8[4]) + JACR8[5] Sin[phi[17]]) (phi[18])'
JACR19[1] -> -((-(ADH32[3, 1] ADH34[2, 1]) + ADH32[2, 1] ADH34[3, 1])
>       (beta[11])') - (-(ADH31[3, 2] ADH34[2, 1]) +
>       ADH31[2, 2] ADH34[3, 1]) (beta[12])' +
>    (ADH34[3, 1] Cos[phi[16]] - ADH34[2, 1] Sin[phi[16]]) (phi[17])' +
>    (-(ADH30[3, 3] ADH34[2, 1]) + ADH30[2, 3] ADH34[3, 1]) (phi[18])'
JACR19[2] -> -((ADH32[3, 1] ADH34[1, 1] - ADH32[1, 1] ADH34[3, 1])
>       (beta[11])') - (ADH31[3, 2] ADH34[1, 1] -
>       ADH23[1, 2] ADH34[3, 1]) (beta[12])' - ADH34[3, 1] (phi[16])' +
>    ADH34[1, 1] Sin[phi[16]] (phi[17])' +
>    (ADH30[3, 3] ADH34[1, 1] - ADH34[3, 1] Sin[phi[17]]) (phi[18])'
JACR19[3] -> -((-(ADH32[2, 1] ADH34[1, 1]) + ADH32[1, 1] ADH34[2, 1])
>       (beta[11])') - (-(ADH31[2, 2] ADH34[1, 1]) +
>       ADH31[2, 2] ADH34[2, 1]) (beta[12])' + ADH34[2, 1] (phi[16])' -
>    ADH34[1, 1] Cos[phi[16]] (phi[17])' +
>    (-(ADH30[2, 3] ADH34[1, 1]) + ADH34[2, 1] Sin[phi[17]]) (phi[18])'
JACR19[4] -> -((-(ADH32[3, 1] JACR9[5]) + ADH32[2, 1] JACR9[6])
>       (beta[11])') - (-(ADH31[3, 2] JACR9[5]) + ADH31[2, 2] JACR9[6])
>     (beta[12])' - (-(ADH34[3, 1] JACR9[5]) + ADH34[2, 1] JACR9[6])
>     (delta[2])' + (Cos[phi[16]] JACR9[6] - JACR9[5] Sin[phi[16]])
>     (phi[17])' + (-(ADH30[3, 3] JACR9[5]) + ADH30[2, 3] JACR9[6])
>     (phi[18])'
JACR19[5] -> -((ADH32[3, 1] JACR9[4] - ADH32[1, 1] JACR9[6])
>       (beta[11])') - (ADH31[3, 2] JACR9[4] - ADH23[1, 2] JACR9[6])
>     (beta[12])' - (ADH34[3, 1] JACR9[4] - ADH34[1, 1] JACR9[6])
>     (delta[2])' - JACR9[6] (phi[16])' +
>    JACR9[4] Sin[phi[16]] (phi[17])' +
>    (ADH30[3, 3] JACR9[4] - JACR9[6] Sin[phi[17]]) (phi[18])'
JACR19[6] -> -((-(ADH32[2, 1] JACR9[4]) + ADH32[1, 1] JACR9[5])
>       (beta[11])') - (-(ADH31[2, 2] JACR9[4]) + ADH23[1, 2] JACR9[5])
>     (beta[12])' - (-(ADH34[2, 1] JACR9[4]) + ADH34[1, 1] JACR9[5])
```

```
>     (delta[2])' + JACR9[5] (phi[16])' -
>    Cos[phi[16]] JACR9[4] (phi[17])' +
>    (-(ADH30[2, 3] JACR9[4]) + JACR9[5] Sin[phi[17]]) (phi[18])'
JACR13[1] -> JACR12[1] (delta[3])' - JACR11[1] (delta[4])'
JACR13[3] -> JACR12[3] (delta[3])' - JACR11[3] (delta[4])'
JACR13[4] -> JACR12[4] (delta[3])' - JACR11[4] (delta[4])' -
>    JACR10[4] (delta[5])'
JACR13[5] -> JACR12[5] (delta[3])' - JACR11[5] (delta[4])'
JACR13[6] -> JACR12[6] (delta[3])' - JACR11[6] (delta[4])' -
>    JACR10[6] (delta[5])'
JACR20[1] -> -(JACR18[1] (beta[11])') - JACR17[1] (beta[12])' -
>    JACR19[1] (delta[2])' + JACR16[1] (phi[18])'
JACR20[2] -> -(JACR18[2] (beta[11])') - JACR17[2] (beta[12])' -
>    JACR19[2] (delta[2])' + JACR15[2] (phi[17])' + JACR16[2] (phi[18])'
JACR20[3] -> -(JACR18[3] (beta[11])') - JACR17[3] (beta[12])' -
>    JACR19[3] (delta[2])' + JACR15[3] (phi[17])' + JACR16[3] (phi[18])'
JACR20[4] -> -(JACR18[4] (beta[11])') - JACR17[4] (beta[12])' -
>    JACR19[4] (delta[2])' + JACR15[4] (phi[17])' + JACR16[4] (phi[18])'
JACR20[5] -> -(JACR18[5] (beta[11])') - JACR17[5] (beta[12])' -
>    JACR19[5] (delta[2])' + JACR14[5] (phi[16])' +
>    JACR15[5] (phi[17])' + JACR16[5] (phi[18])'
JACR20[6] -> -(JACR18[6] (beta[11])') - JACR17[6] (beta[12])' -
>    JACR19[6] (delta[2])' + JACR14[6] (phi[16])' +
>    JACR15[6] (phi[17])' + JACR16[6] (phi[18])'
(beta[11])'' ->
>   (-(JACR13[5] JACR9[4]) + JACR20[5] JACR9[4] + JACR13[4] JACR9[5] -
>      JACR20[4] JACR9[5] - JACR7[5] JACR9[4] (beta[12])'' +
>      JACR7[4] JACR9[5] (beta[12])'' - JACR3[5] JACR9[4] (delta[3])'' +
>      JACR3[4] JACR9[5] (delta[3])'' + JACR2[5] JACR9[4] (delta[4])'' -
>      JACR2[4] JACR9[5] (delta[4])'') /
>    (JACR8[5] JACR9[4] - JACR8[4] JACR9[5])
(delta[2])'' ->
>   -((JACR13[4] - JACR20[4] + JACR8[4] (beta[11])'' +
>       JACR7[4] (beta[12])'' + JACR3[4] (delta[3])'' -
>       JACR2[4] (delta[4])'') / JACR9[4])
(delta[5])'' ->
>   (JACR13[6] - JACR20[6] + JACR8[6] (beta[11])'' +
>      JACR7[6] (beta[12])'' + JACR9[6] (delta[2])'' +
>      JACR3[6] (delta[3])'' - JACR2[6] (delta[4])'') / c1
(phi[17])'' ->
>   (-(ADH30[2, 3] JACR13[3]) - ADH30[3, 3] JACR20[2] +
>      ADH30[2, 3] JACR20[3] + ADH30[3, 3] ADH32[2, 1] (beta[11])'' -
>      ADH30[2, 3] ADH32[3, 1] (beta[11])'' +
>      ADH30[3, 3] ADH31[2, 2] (beta[12])'' -
>      ADH30[2, 3] ADH31[3, 2] (beta[12])'' +
>      ADH30[3, 3] ADH34[2, 1] (delta[2])'' -
>      ADH30[2, 3] ADH34[3, 1] (delta[2])'' -
>      ADH30[2, 3] Sin[delta[5]] (delta[3])'' +
>      ADH30[2, 3] Sin[delta[5]] (delta[4])'' - ADH30[3, 3] (delta[5])'')\
>      / (ADH30[3, 3] Cos[phi[16]] - ADH30[2, 3] Sin[phi[16]])
(phi[18])'' ->
>   (-JACR20[2] + ADH32[2, 1] (beta[11])'' + ADH31[2, 2] (beta[12])'' +
>      ADH34[2, 1] (delta[2])'' - (delta[5])'' - Cos[phi[16]] (phi[17])'')
>      / ADH30[2, 3]
(phi[16])'' ->
>   JACR13[1] - JACR20[1] + ADH32[1, 1] (beta[11])'' +
>    ADH23[1, 2] (beta[12])'' + ADH34[1, 1] (delta[2])'' +
>    Cos[delta[5]] (delta[3])'' - Cos[delta[5]] (delta[4])'' -
>    Sin[phi[17]] (phi[18])''
ADH39[1, 4] -> c1 Cos[delta[5]]
ADH38[1, 4] -> -g + ADH39[1, 4]
                  ADH38[1, 4]
beta[10] -> ArcSin[-----------]
                       d
ADH39[3, 4] -> -(c1 Sin[delta[5]])
ADH38[2, 4] -> ADH2[2, 4] + c2 ADH3[2, 2] + ADH3[2, 3] ADH39[3, 4]
ADH38[3, 4] -> c2 ADH3[3, 2] + ADH3[3, 4] + ADH3[3, 3] ADH39[3, 4]
ADH41[1, 4] -> d Sin[beta[10]]
ADH41[3, 4] -> d Cos[beta[10]]
                          2              2              2
ADH43[1, 1] -> ADH38[1, 4]  + ADH38[2, 4]  + ADH38[3, 4]
                          2              2
ADH42[1, 1] -> ADH41[1, 4]  + ADH41[3, 4]
                  ADH42[1, 1] - ADH43[1, 1] + ADH8[1, 1]
beta[9] -> -ArcSin[-------------------------------------]
                              2 b2 ADH41[3, 4]
ADH45[2, 4] -> -(ADH41[3, 4] Sin[beta[9]])
ADH44[2, 4] -> -b2 + ADH45[2, 4]
ADH45[3, 4] -> ADH41[3, 4] Cos[beta[9]]
delta[1] -> -ArcTan[(ADH38[2, 4] ADH44[2, 4] + ADH38[3, 4] ADH45[3, 4]) /
                    2              2
>      (ADH38[2, 4]  + ADH38[3, 4] ),
>     (-(ADH38[3, 4] ADH44[2, 4]) + ADH38[2, 4] ADH45[3, 4]) /
                    2              2
>      (ADH38[2, 4]  + ADH38[3, 4] )]
ADH49[3, 2] -> -(Cos[delta[1]] Sin[beta[9]]) - Cos[beta[9]] Sin[delta[1]]
ADH49[3, 3] -> Cos[beta[9]] Cos[delta[1]] - Sin[beta[9]] Sin[delta[1]]
ADH48[1, 2] -> -(ADH49[3, 2] Sin[beta[10]])
ADH48[1, 3] -> -(ADH49[3, 3] Sin[beta[10]])
ADH49[2, 2] -> Cos[beta[9]] Cos[delta[1]] - Sin[beta[9]] Sin[delta[1]]
ADH49[2, 3] -> Cos[delta[1]] Sin[beta[9]] + Cos[beta[9]] Sin[delta[1]]
ADH46[1, 1] ->
>   Cos[beta[10]] Cos[delta[5]] -
>    (ADH3[2, 3] ADH48[1, 2] + ADH3[3, 3] ADH48[1, 3]) Sin[delta[5]]
ADH46[2, 1] ->
>   -((ADH3[2, 3] ADH49[2, 2] + ADH3[3, 3] ADH49[2, 3]) Sin[delta[5]])
                             ADH46[2, 1]
phi[15] -> -3.14159 - ArcTan[-----------]
                             ADH46[1, 1]
ADH48[3, 2] -> ADH49[3, 2] Cos[beta[10]]
ADH48[3, 3] -> ADH49[3, 3] Cos[beta[10]]
ADH46[3, 1] ->
>   Cos[delta[5]] Sin[beta[10]] -
>    (ADH3[2, 3] ADH48[3, 2] + ADH3[3, 3] ADH48[3, 3]) Sin[delta[5]]
ADH53[1, 1] -> ADH46[1, 1] Cos[phi[15]] - ADH46[2, 1] Sin[phi[15]]
                          ADH53[1, 1]
phi[14] -> ArcTan[--------------------------,
                                2             2
                    ADH46[3, 1]  + ADH53[1, 1]
               ADH46[3, 1]
>    ---------------------------]
                  2             2
     ADH46[3, 1]  + ADH53[1, 1]
ADH46[1, 2] -> ADH3[2, 2] ADH48[1, 2] + ADH3[3, 2] ADH48[1, 3]
ADH46[2, 2] -> ADH3[2, 2] ADH49[2, 2] + ADH3[3, 2] ADH49[2, 3]
ADH46[3, 2] -> ADH3[2, 2] ADH48[3, 2] + ADH3[3, 2] ADH48[3, 3]
ADH55[3, 1] -> -(Cos[phi[15]] Sin[phi[14]])
ADH55[3, 2] -> Sin[phi[14]] Sin[phi[15]]
ADH54[2, 2] -> ADH46[2, 2] Cos[phi[15]] + ADH46[1, 2] Sin[phi[15]]
ADH54[3, 2] ->
>   ADH46[1, 2] ADH55[3, 1] + ADH46[2, 2] ADH55[3, 2] +
>    ADH46[3, 2] Cos[phi[14]]
phi[13] -> -ArcTan[ADH54[2, 2], ADH54[3, 2]]
ADH72[2, 4] -> -(b2 Cos[delta[1]])
ADH72[3, 4] -> -(b2 Sin[delta[1]])
ADH67[2, 4] -> h2 Cos[delta[3]]
ADH67[3, 4] -> h2 Sin[delta[3]]
ADH71[2, 4] ->
>   h2 Cos[delta[3]] + ADH72[2, 4] Cos[delta[3]] -
>    ADH72[3, 4] Sin[delta[3]]
ADH71[3, 4] ->
>   ADH72[3, 4] Cos[delta[3]] + h2 Sin[delta[3]] +
>    ADH72[2, 4] Sin[delta[3]]
ADH59[2, 4] -> -(c2 Cos[phi[13]])
ADH59[3, 4] -> c2 Sin[phi[13]]
ADH63[1, 4] -> -(ADH29[3, 4] Sin[delta[5]])
ADH63[3, 4] -> ADH29[3, 4] Cos[delta[5]]
ADH64[2, 4] -> f + ADH29[2, 4]
ADH66[1, 4] ->
>   g Cos[delta[5]] - ADH29[3, 4] Sin[delta[5]] -
>    ADH67[3, 4] Cos[delta[4]] Sin[delta[5]] +
>    ADH67[2, 4] Sin[delta[4]] Sin[delta[5]]
ADH66[3, 4] ->
>   ADH29[3, 4] Cos[delta[5]] + ADH67[3, 4] Cos[delta[4]] Cos[delta[5]] -
>    ADH67[2, 4] Cos[delta[5]] Sin[delta[4]] + g Sin[delta[5]]
ADH70[1, 4] ->
>   g Cos[delta[5]] - ADH29[3, 4] Sin[delta[5]] -
>    ADH71[3, 4] Cos[delta[4]] Sin[delta[5]] +
>    ADH71[2, 4] Sin[delta[4]] Sin[delta[5]]
ADH70[3, 4] ->
>   ADH29[3, 4] Cos[delta[5]] + ADH71[3, 4] Cos[delta[4]] Cos[delta[5]] -
>    ADH71[2, 4] Cos[delta[5]] Sin[delta[4]] + g Sin[delta[5]]
ADH58[1, 4] -> -(c1 Cos[phi[14]]) - ADH59[3, 4] Sin[phi[14]]
ADH66[2, 4] ->
>   ADH64[2, 4] + ADH67[2, 4] Cos[delta[4]] + ADH67[3, 4] Sin[delta[4]]
ADH70[2, 4] ->
>   ADH64[2, 4] + ADH71[2, 4] Cos[delta[4]] + ADH71[3, 4] Sin[delta[4]]
ADH57[1, 4] -> ADH58[1, 4] Cos[phi[15]] + ADH59[2, 4] Sin[phi[15]]
ADH57[2, 4] -> ADH59[2, 4] Cos[phi[15]] - ADH58[1, 4] Sin[phi[15]]
ADH58[1, 2] -> Sin[phi[13]] Sin[phi[14]]
ADH58[1, 3] -> -(Cos[phi[13]] Sin[phi[14]])
ADH58[3, 3] -> Cos[phi[13]] Cos[phi[14]]
ADH58[3, 4] -> ADH59[3, 4] Cos[phi[14]] - c1 Sin[phi[14]]
ADH61[1, 4] ->
>   ADH58[1, 4] Cos[phi[15]] + ADH59[2, 4] Sin[phi[15]] +
>    f (ADH58[1, 2] Cos[phi[15]] + Cos[phi[13]] Sin[phi[15]])
ADH61[2, 4] ->
>   ADH59[2, 4] Cos[phi[15]] - ADH58[1, 4] Sin[phi[15]] +
>    f (Cos[phi[13]] Cos[phi[15]] - ADH58[1, 2] Sin[phi[15]])
ADH58[3, 2] -> -(Cos[phi[14]] Sin[phi[13]])
ADH61[3, 4] -> f ADH58[3, 2] + ADH58[3, 4]
ADH62[1, 4] ->
```

```
>   ADH58[i, 4] Cos[phi[15]] + ADH63[i, 4] Cos[phi[14]] Cos[phi[15]] +
>    ADH59[2, 4] Sin[phi[15]] +
>    ADH64[2, 4] (ADH58[i, 2] Cos[phi[15]] + Cos[phi[13]] Sin[phi[15]]) +
>    ADH63[3, 4] (ADH58[i, 3] Cos[phi[15]] + Sin[phi[13]] Sin[phi[15]])
ADH62[2, 4] ->
>   ADH59[2, 4] Cos[phi[15]] - ADH58[i, 4] Sin[phi[15]] -
>    ADH63[i, 4] Cos[phi[14]] Sin[phi[15]] +
>    ADH64[2, 4] (Cos[phi[13]] Cos[phi[15]] - ADH58[i, 2] Sin[phi[15]]) +
>    ADH63[3, 4] (Cos[phi[15]] Sin[phi[13]] - ADH58[i, 3] Sin[phi[15]])
ADH62[3, 4] ->
>   ADH58[3, 4] + ADH58[3, 3] ADH63[3, 4] + ADH58[3, 2] ADH64[2, 4] +
>    ADH63[i, 4] Sin[phi[14]]
ADH65[i, 4] ->
>   ADH58[i, 4] Cos[phi[15]] + ADH66[i, 4] Cos[phi[14]] Cos[phi[15]] +
>    ADH59[2, 4] Sin[phi[15]] +
>    ADH66[2, 4] (ADH58[i, 2] Cos[phi[15]] + Cos[phi[13]] Sin[phi[15]]) +
>    ADH66[3, 4] (ADH58[i, 3] Cos[phi[15]] + Sin[phi[13]] Sin[phi[15]])
ADH65[2, 4] ->
>   ADH59[2, 4] Cos[phi[15]] - ADH58[i, 4] Sin[phi[15]] -
>    ADH66[i, 4] Cos[phi[14]] Sin[phi[15]] +
>    ADH66[2, 4] (Cos[phi[13]] Cos[phi[15]] - ADH58[i, 2] Sin[phi[15]]) +
>    ADH66[3, 4] (Cos[phi[15]] Sin[phi[13]] - ADH58[i, 3] Sin[phi[15]])
ADH65[3, 4] ->
>   ADH58[3, 4] + ADH58[3, 2] ADH66[2, 4] + ADH58[3, 3] ADH66[3, 4] +
>    ADH66[i, 4] Sin[phi[14]]
ADH69[i, 4] ->
>   ADH58[i, 4] Cos[phi[15]] + ADH70[i, 4] Cos[phi[14]] Cos[phi[15]] +
>    ADH59[2, 4] Sin[phi[15]] +
>    ADH70[2, 4] (ADH58[i, 2] Cos[phi[15]] + Cos[phi[13]] Sin[phi[15]]) +
>    ADH70[3, 4] (ADH58[i, 3] Cos[phi[15]] + Sin[phi[13]] Sin[phi[15]])
ADH69[2, 4] ->
>   ADH59[2, 4] Cos[phi[15]] - ADH58[i, 4] Sin[phi[15]] -
>    ADH70[i, 4] Cos[phi[14]] Sin[phi[15]] +
>    ADH70[2, 4] (Cos[phi[13]] Cos[phi[15]] - ADH58[i, 2] Sin[phi[15]]) +
>    ADH70[3, 4] (Cos[phi[15]] Sin[phi[13]] - ADH58[i, 3] Sin[phi[15]])
ADH69[3, 4] ->
>   ADH58[3, 4] + ADH58[3, 2] ADH70[2, 4] + ADH58[3, 3] ADH70[3, 4] +
>    ADH70[i, 4] Sin[phi[14]]
ADH56[i, i] -> Cos[phi[14]] Cos[phi[15]]
ADH56[2, i] -> -(Cos[phi[14]] Sin[phi[15]])
ADH57[i, 2] -> ADH58[i, 2] Cos[phi[15]] + Cos[phi[13]] Sin[phi[15]]
ADH57[2, 2] -> Cos[phi[13]] Cos[phi[15]] - ADH58[i, 2] Sin[phi[15]]
ADH61[i, i] ->
>   Cos[delta[5]] Cos[phi[14]] Cos[phi[15]] +
>    Sin[delta[5]] (ADH58[i, 3] Cos[phi[15]] + Sin[phi[13]] Sin[phi[15]])
ADH61[2, i] ->
>   -(Cos[delta[5]] Cos[phi[14]] Sin[phi[15]]) +
>    Sin[delta[5]] (Cos[phi[15]] Sin[phi[13]] - ADH58[i, 3] Sin[phi[15]])
ADH61[3, i] -> ADH58[3, 3] Sin[delta[5]] + Cos[delta[5]] Sin[phi[14]]
ADH62[i, i] ->
>   Cos[delta[5]] Cos[phi[14]] Cos[phi[15]] +
>    Sin[delta[5]] (ADH58[i, 3] Cos[phi[15]] + Sin[phi[13]] Sin[phi[15]])
ADH62[2, i] ->
>   -(Cos[delta[5]] Cos[phi[14]] Sin[phi[15]]) +
>    Sin[delta[5]] (Cos[phi[15]] Sin[phi[13]] - ADH58[i, 3] Sin[phi[15]])
ADH62[3, i] -> ADH58[3, 3] Sin[delta[5]] + Cos[delta[5]] Sin[phi[14]]
ADH65[i, i] ->
>   Cos[delta[5]] Cos[phi[14]] Cos[phi[15]] +
>    Sin[delta[5]] (ADH58[i, 3] Cos[phi[15]] + Sin[phi[13]] Sin[phi[15]])
ADH65[2, i] ->
>   -(Cos[delta[5]] Cos[phi[14]] Sin[phi[15]]) +
>    Sin[delta[5]] (Cos[phi[15]] Sin[phi[13]] - ADH58[i, 3] Sin[phi[15]])
ADH65[3, i] -> ADH58[3, 3] Sin[delta[5]] + Cos[delta[5]] Sin[phi[14]]
ADH69[i, i] ->
>   Cos[delta[5]] Cos[phi[14]] Cos[phi[15]] +
>    Sin[delta[5]] (ADH58[i, 3] Cos[phi[15]] + Sin[phi[13]] Sin[phi[15]])
ADH69[2, i] ->
>   -(Cos[delta[5]] Cos[phi[14]] Sin[phi[15]]) +
>    Sin[delta[5]] (Cos[phi[15]] Sin[phi[13]] - ADH58[i, 3] Sin[phi[15]])
ADH69[3, i] -> ADH58[3, 3] Sin[delta[5]] + Cos[delta[5]] Sin[phi[14]]
JACR25[4] -> ADH57[2, 4] ADH58[3, 2] - ADH57[2, 2] ADH58[3, 4]
JACR25[5] -> -(ADH57[i, 4] ADH58[3, 2]) + ADH57[i, 2] ADH58[3, 4]
JACR25[6] -> ADH57[i, 4] ADH57[2, 2] - ADH57[i, 2] ADH57[2, 4]
JACR26[4] -> ADH61[2, 4] ADH61[3, i] - ADH61[2, i] ADH61[3, 4]
JACR26[5] -> -(ADH61[i, 4] ADH61[3, i]) + ADH61[i, i] ADH61[3, 4]
JACR26[6] -> ADH61[i, 4] ADH61[2, i] - ADH61[i, i] ADH61[2, 4]
JACR27[4] -> ADH62[2, 4] ADH62[3, i] - ADH62[2, i] ADH62[3, 4]
JACR27[5] -> -(ADH62[i, 4] ADH62[3, i]) + ADH62[i, i] ADH62[3, 4]
JACR27[6] -> ADH62[i, 4] ADH62[2, i] - ADH62[i, i] ADH62[2, 4]
JACR28[4] -> ADH65[2, 4] ADH65[3, i] - ADH65[2, i] ADH65[3, 4]
JACR28[5] -> -(ADH65[i, 4] ADH65[3, i]) + ADH65[i, i] ADH65[3, 4]
JACR28[6] -> ADH65[i, 4] ADH65[2, i] - ADH65[i, i] ADH65[2, 4]
JACR29[4] -> ADH69[2, 4] ADH69[3, i] - ADH69[2, i] ADH69[3, 4]
JACR29[5] -> -(ADH69[i, 4] ADH69[3, i]) + ADH69[i, i] ADH69[3, 4]
JACR29[6] -> ADH69[i, 4] ADH69[2, i] - ADH69[i, i] ADH69[2, 4]
(beta[9])' -> (JACR27[6] JACR28[5] (delta[3])' -
>      JACR27[5] JACR28[6] (delta[3])' -
>      JACR26[6] JACR28[5] (delta[4])' +
>      JACR26[5] JACR28[6] (delta[4])' -
>      JACR25[6] JACR28[5] (delta[5])' + JACR25[5] JACR28[6] (delta[5])')\
>      / (JACR28[6] JACR29[5] - JACR28[5] JACR29[6])
(delta[i])' ->
>   (-(JACR29[5] (beta[9])') - JACR27[5] (delta[3])' +
>      JACR26[5] (delta[4])' + JACR25[5] (delta[5])') / JACR28[5]
(beta[10])' ->
>   (-(JACR29[4] (beta[9])') - JACR28[4] (delta[1])' -
>      JACR27[4] (delta[3])' + JACR26[4] (delta[4])' +
>      JACR25[4] (delta[5])') / d
(phi[13])' -> (ADH69[i, i] Cos[phi[15]] (beta[9])' -
>      ADH69[2, i] Sin[phi[15]] (beta[9])' - Sin[phi[15]] (beta[10])' +
>      ADH65[i, i] Cos[phi[15]] (delta[i])' -
>      ADH65[2, i] Sin[phi[15]] (delta[i])' +
>      ADH62[i, i] Cos[phi[15]] (delta[3])' -
>      ADH62[2, i] Sin[phi[15]] (delta[3])' -
>      ADH61[i, i] Cos[phi[15]] (delta[4])' +
>      ADH61[2, i] Sin[phi[15]] (delta[4])' -
>      ADH57[i, 2] Cos[phi[15]] (delta[5])' +
>      ADH57[2, 2] Sin[phi[15]] (delta[5])') /
>    (ADH56[i, i] Cos[phi[15]] - ADH56[2, i] Sin[phi[15]])
(phi[14])' -> -(Csc[phi[15]] (-(ADH69[i, i] (beta[9])') -
>       ADH65[i, i] (delta[i])' - ADH62[i, i] (delta[3])' +
>       ADH61[i, i] (delta[4])' + ADH57[i, 2] (delta[5])' +
>       ADH56[i, i] (phi[13])'))
(phi[15])' -> ADH69[3, i] (beta[9])' + ADH65[3, i] (delta[i])' +
>    ADH62[3, i] (delta[3])' - ADH61[3, i] (delta[4])' -
>    ADH58[3, 2] (delta[5])' - Sin[phi[14]] (phi[13])'
JACR30[6] -> d (beta[10])'
JACR32[4] -> -(JACR29[5] (beta[9])') - JACR28[5] (delta[i])' -
>    JACR27[5] (delta[3])' + JACR26[5] (delta[4])' + JACR25[5] (delta[5])'
JACR32[5] -> JACR29[4] (beta[9])' + JACR28[4] (delta[i])' +
>    JACR27[4] (delta[3])' - JACR26[4] (delta[4])' - JACR25[4] (delta[5])'
JACR33[i] -> Cos[phi[15]] (phi[15])'
JACR33[2] -> -(Sin[phi[15]] (phi[15])')
JACR33[4] -> Cos[phi[15]] JACR29[6] (beta[9])' +
>    Cos[phi[15]] JACR28[6] (delta[i])' +
>    Cos[phi[15]] JACR27[6] (delta[3])' -
>    Cos[phi[15]] JACR26[6] (delta[4])' -
>    Cos[phi[15]] JACR25[6] (delta[5])'
JACR33[5] -> -(JACR29[6] Sin[phi[15]] (beta[9])') -
>    JACR28[6] Sin[phi[15]] (delta[i])' -
>    JACR27[6] Sin[phi[15]] (delta[3])' +
>    JACR26[6] Sin[phi[15]] (delta[4])' +
>    JACR25[6] Sin[phi[15]] (delta[5])'
JACR33[6] -> (-(Cos[phi[15]] JACR29[4]) + JACR29[5] Sin[phi[15]])
>     (beta[9])' + (-(Cos[phi[15]] JACR28[4]) + JACR28[5] Sin[phi[15]])
>     (delta[i])' + (-(Cos[phi[15]] JACR27[4]) + JACR27[5] Sin[phi[15]])
>     (delta[3])' - (-(Cos[phi[15]] JACR26[4]) + JACR26[5] Sin[phi[15]])
>     (delta[4])' - (-(Cos[phi[15]] JACR25[4]) + JACR25[5] Sin[phi[15]])
>     (delta[5])'
JACR34[1] -> -(Cos[phi[15]] Sin[phi[14]] (phi[14])') +
>    ADH56[2, i] (phi[15])'
JACR34[2] -> Sin[phi[14]] Sin[phi[15]] (phi[14])' - ADH56[i, i] (phi[15])'
JACR34[3] -> -((-(ADH56[i, i] Cos[phi[15]]) + ADH56[2, i] Sin[phi[15]])
>      (phi[14])')
JACR34[4] -> (ADH56[2, i] JACR29[6] - JACR29[5] Sin[phi[14]])
>     (beta[9])' + (ADH56[2, i] JACR28[6] - JACR28[5] Sin[phi[14]])
>     (delta[i])' + (ADH56[2, i] JACR27[6] - JACR27[5] Sin[phi[14]])
>     (delta[3])' - (ADH56[2, i] JACR26[6] - JACR26[5] Sin[phi[14]])
>     (delta[4])' - (ADH56[2, i] JACR25[6] - JACR25[5] Sin[phi[14]])
>     (delta[5])'
JACR34[5] -> (-(ADH56[i, i] JACR29[6]) + JACR29[4] Sin[phi[14]])
>     (beta[9])' + (-(ADH56[i, i] JACR28[6]) + JACR28[4] Sin[phi[14]])
>     (delta[i])' + (-(ADH56[i, i] JACR27[6]) + JACR27[4] Sin[phi[14]])
>     (delta[3])' - (-(ADH56[i, i] JACR26[6]) + JACR26[4] Sin[phi[14]])
>     (delta[4])' - (-(ADH56[i, i] JACR25[6]) + JACR25[4] Sin[phi[14]])
>     (delta[5])'
JACR34[6] -> (-(ADH56[2, i] JACR29[4]) + ADH56[i, i] JACR29[5])
>     (beta[9])' + (-(ADH56[2, i] JACR28[4]) + ADH56[i, i] JACR28[5])
>     (delta[i])' + (-(ADH56[2, i] JACR27[4]) + ADH56[i, i] JACR27[5])
>     (delta[3])' - (-(ADH56[2, i] JACR26[4]) + ADH56[i, i] JACR26[5])
>     (delta[4])' - (-(ADH56[2, i] JACR25[4]) + ADH56[i, i] JACR25[5])
>     (delta[5])'
JACR35[1] -> -((ADH56[2, i] ADH58[3, 2] - ADH57[2, 2] Sin[phi[14]])
>       (phi[13])') - ADH58[3, 2] Cos[phi[15]] (phi[14])' +
>    ADH57[2, 2] (phi[15])'
JACR35[2] -> -((-(ADH56[i, i] ADH58[3, 2]) + ADH57[i, 2] Sin[phi[14]])
>       (phi[13])') + ADH58[3, 2] Sin[phi[15]] (phi[14])' -
>    ADH57[i, 2] (phi[15])'
JACR35[3] -> -((-(ADH56[2, i] ADH57[i, 2]) + ADH56[i, i] ADH57[2, 2])
>       (phi[13])') - (-(ADH57[i, 2] Cos[phi[15]]) +
>       ADH57[2, 2] Sin[phi[15]]) (phi[14])'
JACR35[4] -> (-(ADH58[3, 2] JACR29[5]) + ADH57[2, 2] JACR29[6])
```

```
>     (beta[9])' + (-(ADH58[3, 2] JACR28[5]) + ADH57[2, 2] JACR28[6])
>     (delta[1])' + (-(ADH58[3, 2] JACR27[5]) + ADH57[2, 2] JACR27[6])
>     (delta[3])' - (-(ADH58[3, 2] JACR26[5]) + ADH57[2, 2] JACR26[6])
>     (delta[4])' - (-(ADH58[3, 2] JACR25[5]) + ADH57[2, 2] JACR25[6])
>     (delta[5])' - (ADH56[2, 1] JACR25[6] - JACR25[5] Sin[phi[14]])
>     (phi[13])' - Cos[phi[15]] JACR25[6] (phi[14])' +
>   JACR25[5] (phi[15])'
JACR35[5] -> (ADH58[3, 2] JACR29[4] - ADH57[1, 2] JACR29[6]) (beta[9])' +
>   (ADH58[3, 2] JACR28[4] - ADH57[1, 2] JACR28[6]) (delta[1])' +
>   (ADH58[3, 2] JACR27[4] - ADH57[1, 2] JACR27[6]) (delta[3])' -
>   (ADH58[3, 2] JACR26[4] - ADH57[1, 2] JACR26[6]) (delta[4])' -
>   (ADH58[3, 2] JACR25[4] - ADH57[1, 2] JACR25[6]) (delta[5])' -
>   (-(ADH56[1, 1] JACR25[6]) + JACR25[4] Sin[phi[14]]) (phi[13])' +
>   JACR25[6] Sin[phi[15]] (phi[14])' - JACR25[4] (phi[15])'
JACR35[6] -> (-(ADH57[2, 2] JACR29[4]) + ADH57[1, 2] JACR29[5])
>     (beta[9])' + (-(ADH57[2, 2] JACR28[4]) + ADH57[1, 2] JACR28[5])
>     (delta[1])' + (-(ADH57[2, 2] JACR27[4]) + ADH57[1, 2] JACR27[5])
>     (delta[3])' - (-(ADH57[2, 2] JACR26[4]) + ADH57[1, 2] JACR26[5])
>     (delta[4])' - (-(ADH57[2, 2] JACR25[4]) + ADH57[1, 2] JACR25[5])
>     (delta[5])' - (-(ADH56[2, 1] JACR25[4]) + ADH56[1, 1] JACR25[5])
>     (phi[13])' - (-(Cos[phi[15]] JACR25[4]) + JACR25[5] Sin[phi[15]])
>     (phi[14])'
JACR36[1] -> -((-(ADH58[3, 2] ADH61[2, 1]) + ADH57[2, 2] ADH61[3, 1])
>       (delta[5])') - (ADH56[2, 1] ADH61[3, 1] -
>       ADH61[2, 1] Sin[phi[14]]) (phi[13])' -
>   ADH61[3, 1] Cos[phi[15]] (phi[14])' + ADH61[2, 1] (phi[15])'
JACR36[2] -> -((ADH58[3, 2] ADH61[1, 1] - ADH57[1, 2] ADH61[3, 1])
>       (delta[5])') - (-(ADH56[1, 1] ADH61[3, 1]) +
>       ADH61[1, 1] Sin[phi[14]]) (phi[13])' +
>   ADH61[3, 1] Sin[phi[15]] (phi[14])' - ADH61[1, 1] (phi[15])'
JACR36[3] -> -((-(ADH57[2, 2] ADH61[1, 1]) + ADH57[1, 2] ADH61[2, 1])
>       (delta[5])') - (-(ADH56[2, 1] ADH61[1, 1]) +
>       ADH56[1, 1] ADH61[2, 1]) (phi[13])' -
>   (-(ADH61[1, 1] Cos[phi[15]]) + ADH61[2, 1] Sin[phi[15]]) (phi[14])'
JACR36[4] -> (-(ADH61[3, 1] JACR29[5]) + ADH61[2, 1] JACR29[6])
>     (beta[9])' + (-(ADH61[3, 1] JACR28[5]) + ADH61[2, 1] JACR28[6])
>     (delta[1])' + (-(ADH61[3, 1] JACR27[5]) + ADH61[2, 1] JACR27[6])
>     (delta[3])' - (-(ADH61[3, 1] JACR26[5]) + ADH61[2, 1] JACR26[6])
>     (delta[4])' - (-(ADH58[3, 2] JACR26[5]) + ADH57[2, 2] JACR26[6])
>     (delta[5])' - (ADH56[2, 1] JACR26[6] - JACR26[5] Sin[phi[14]])
>     (phi[13])' - Cos[phi[15]] JACR26[6] (phi[14])' +
>   JACR26[5] (phi[15])'
JACR36[5] -> (ADH61[3, 1] JACR29[4] - ADH61[1, 1] JACR29[6]) (beta[9])' +
>   (ADH61[3, 1] JACR28[4] - ADH61[1, 1] JACR28[6]) (delta[1])' +
>   (ADH61[3, 1] JACR27[4] - ADH61[1, 1] JACR27[6]) (delta[3])' -
>   (ADH61[3, 1] JACR26[4] - ADH61[1, 1] JACR26[6]) (delta[4])' -
>   (ADH58[3, 2] JACR26[4] - ADH57[1, 2] JACR26[6]) (delta[5])' -
>   (-(ADH56[1, 1] JACR26[6]) + JACR26[4] Sin[phi[14]]) (phi[13])' +
>   JACR26[6] Sin[phi[15]] (phi[14])' - JACR26[4] (phi[15])'
JACR36[6] -> (-(ADH61[2, 1] JACR29[4]) + ADH61[1, 1] JACR29[5])
>     (beta[9])' + (-(ADH61[2, 1] JACR28[4]) + ADH61[1, 1] JACR28[5])
>     (delta[1])' + (-(ADH61[2, 1] JACR27[4]) + ADH61[1, 1] JACR27[5])
>     (delta[3])' - (-(ADH61[2, 1] JACR26[4]) + ADH61[1, 1] JACR26[5])
>     (delta[4])' - (-(ADH57[2, 2] JACR26[4]) + ADH57[1, 2] JACR26[5])
>     (delta[5])' - (-(ADH56[2, 1] JACR26[4]) + ADH56[1, 1] JACR26[5])
>     (phi[13])' - (-(Cos[phi[15]] JACR26[4]) + JACR26[5] Sin[phi[15]])
>     (phi[14])'
JACR37[1] -> -((-(ADH61[3, 1] ADH62[2, 1]) + ADH61[2, 1] ADH62[3, 1])
>       (delta[4])') - (-(ADH58[3, 2] ADH62[2, 1]) +
>       ADH57[2, 2] ADH62[3, 1]) (delta[5])' -
>   (ADH56[2, 1] ADH62[3, 1] - ADH62[2, 1] Sin[phi[14]]) (phi[13])' -
>   ADH62[3, 1] Cos[phi[15]] (phi[14])' + ADH62[2, 1] (phi[15])'
JACR37[2] -> -((ADH61[3, 1] ADH62[1, 1] - ADH61[1, 1] ADH62[3, 1])
>       (delta[4])') - (ADH58[3, 2] ADH62[1, 1] -
>       ADH57[1, 2] ADH62[3, 1]) (delta[5])' -
>   (-(ADH56[1, 1] ADH62[3, 1]) + ADH62[1, 1] Sin[phi[14]]) (phi[13])' +
>   ADH62[3, 1] Sin[phi[15]] (phi[14])' - ADH62[1, 1] (phi[15])'
JACR37[3] -> -((-(ADH61[2, 1] ADH62[1, 1]) + ADH61[1, 1] ADH62[2, 1])
>       (delta[4])') - (-(ADH57[2, 2] ADH62[1, 1]) +
>       ADH57[1, 2] ADH62[2, 1]) (delta[5])' -
>   (-(ADH56[2, 1] ADH62[1, 1]) + ADH56[1, 1] ADH62[2, 1]) (phi[13])' -
>   (-(ADH62[1, 1] Cos[phi[15]]) + ADH62[2, 1] Sin[phi[15]]) (phi[14])'
JACR37[4] -> (-(ADH62[3, 1] JACR29[5]) + ADH62[2, 1] JACR29[6])
>     (beta[9])' + (-(ADH62[3, 1] JACR28[5]) + ADH62[2, 1] JACR28[6])
>     (delta[1])' + (-(ADH62[3, 1] JACR27[5]) + ADH62[2, 1] JACR27[6])
>     (delta[3])' - (-(ADH61[3, 1] JACR27[5]) + ADH61[2, 1] JACR27[6])
>     (delta[4])' - (-(ADH58[3, 2] JACR27[5]) + ADH57[2, 2] JACR27[6])
>     (delta[5])' - (ADH56[2, 1] JACR27[6] - JACR27[5] Sin[phi[14]])
>     (phi[13])' - Cos[phi[15]] JACR27[6] (phi[14])' +
>   JACR27[5] (phi[15])'
JACR37[5] -> (ADH62[3, 1] JACR29[4] - ADH62[1, 1] JACR29[6]) (beta[9])' +
>   (ADH62[3, 1] JACR28[4] - ADH62[1, 1] JACR28[6]) (delta[1])' +
>   (ADH62[3, 1] JACR27[4] - ADH62[1, 1] JACR27[6]) (delta[3])' -
>   (ADH61[3, 1] JACR27[4] - ADH61[1, 1] JACR27[6]) (delta[4])' -
>   (ADH58[3, 2] JACR27[4] - ADH57[1, 2] JACR27[6]) (delta[5])' -
>   (-(ADH56[1, 1] JACR27[6]) + JACR27[4] Sin[phi[14]]) (phi[13])' +
>   JACR27[6] Sin[phi[15]] (phi[14])' - JACR27[4] (phi[15])'
JACR37[6] -> (-(ADH62[2, 1] JACR29[4]) + ADH62[1, 1] JACR29[5])
>     (beta[9])' + (-(ADH62[2, 1] JACR28[4]) + ADH62[1, 1] JACR28[5])
>     (delta[1])' + (-(ADH62[2, 1] JACR27[4]) + ADH62[1, 1] JACR27[5])
>     (delta[3])' - (-(ADH61[2, 1] JACR27[4]) + ADH61[1, 1] JACR27[5])
>     (delta[4])' - (-(ADH57[2, 2] JACR27[4]) + ADH57[1, 2] JACR27[5])
>     (delta[5])' - (-(ADH56[2, 1] JACR27[4]) + ADH56[1, 1] JACR27[5])
>     (phi[13])' - (-(Cos[phi[15]] JACR27[4]) + JACR27[5] Sin[phi[15]])
>     (phi[14])'
JACR38[1] -> (-(ADH62[3, 1] ADH65[2, 1]) + ADH62[2, 1] ADH65[3, 1])
>     (delta[3])' - (-(ADH61[3, 1] ADH65[2, 1]) +
>       ADH61[2, 1] ADH65[3, 1]) (delta[4])' -
>   (-(ADH58[3, 2] ADH65[2, 1]) + ADH57[2, 2] ADH65[3, 1]) (delta[5])' -
>   (ADH56[2, 1] ADH65[3, 1] - ADH65[2, 1] Sin[phi[14]]) (phi[13])' -
>   ADH65[3, 1] Cos[phi[15]] (phi[14])' + ADH65[2, 1] (phi[15])'
JACR38[2] -> (ADH62[3, 1] ADH65[1, 1] - ADH62[1, 1] ADH65[3, 1])
>     (delta[3])' - (ADH61[3, 1] ADH65[1, 1] - ADH61[1, 1] ADH65[3, 1])
>     (delta[4])' - (ADH58[3, 2] ADH65[1, 1] - ADH57[1, 2] ADH65[3, 1])
>     (delta[5])' - (-(ADH56[1, 1] ADH65[3, 1]) +
>       ADH65[1, 1] Sin[phi[14]]) (phi[13])' +
>   ADH65[3, 1] Sin[phi[15]] (phi[14])' - ADH65[1, 1] (phi[15])'
JACR38[3] -> (-(ADH62[2, 1] ADH65[1, 1]) + ADH62[1, 1] ADH65[2, 1])
>     (delta[3])' - (-(ADH61[2, 1] ADH65[1, 1]) +
>       ADH61[1, 1] ADH65[2, 1]) (delta[4])' -
>   (-(ADH57[2, 2] ADH65[1, 1]) + ADH57[1, 2] ADH65[2, 1]) (delta[5])' -
>   (-(ADH56[2, 1] ADH65[1, 1]) + ADH56[1, 1] ADH65[2, 1]) (phi[13])' -
>   (-(ADH65[1, 1] Cos[phi[15]]) + ADH65[2, 1] Sin[phi[15]]) (phi[14])'
JACR38[4] -> (-(ADH65[3, 1] JACR29[5]) + ADH65[2, 1] JACR29[6])
>     (beta[9])' + (-(ADH65[3, 1] JACR28[5]) + ADH65[2, 1] JACR28[6])
>     (delta[1])' + (-(ADH62[3, 1] JACR28[5]) + ADH62[2, 1] JACR28[6])
>     (delta[3])' - (-(ADH61[3, 1] JACR28[5]) + ADH61[2, 1] JACR28[6])
>     (delta[4])' - (-(ADH58[3, 2] JACR28[5]) + ADH57[2, 2] JACR28[6])
>     (delta[5])' - (ADH56[2, 1] JACR28[6] - JACR28[5] Sin[phi[14]])
>     (phi[13])' - Cos[phi[15]] JACR28[6] (phi[14])' +
>   JACR28[5] (phi[15])'
JACR38[5] -> (ADH65[3, 1] JACR29[4] - ADH65[1, 1] JACR29[6]) (beta[9])' +
>   (ADH65[3, 1] JACR28[4] - ADH65[1, 1] JACR28[6]) (delta[1])' +
>   (ADH62[3, 1] JACR28[4] - ADH62[1, 1] JACR28[6]) (delta[3])' -
>   (ADH61[3, 1] JACR28[4] - ADH61[1, 1] JACR28[6]) (delta[4])' -
>   (ADH58[3, 2] JACR28[4] - ADH57[1, 2] JACR28[6]) (delta[5])' -
>   (-(ADH56[1, 1] JACR28[6]) + JACR28[4] Sin[phi[14]]) (phi[13])' +
>   JACR28[6] Sin[phi[15]] (phi[14])' - JACR28[4] (phi[15])'
JACR38[6] -> (-(ADH65[2, 1] JACR29[4]) + ADH65[1, 1] JACR29[5])
>     (beta[9])' + (-(ADH65[2, 1] JACR28[4]) + ADH65[1, 1] JACR28[5])
>     (delta[1])' + (-(ADH62[2, 1] JACR28[4]) + ADH62[1, 1] JACR28[5])
>     (delta[3])' - (-(ADH61[2, 1] JACR28[4]) + ADH61[1, 1] JACR28[5])
>     (delta[4])' - (-(ADH57[2, 2] JACR28[4]) + ADH57[1, 2] JACR28[5])
>     (delta[5])' - (-(ADH56[2, 1] JACR28[4]) + ADH56[1, 1] JACR28[5])
>     (phi[13])' - (-(Cos[phi[15]] JACR28[4]) + JACR28[5] Sin[phi[15]])
>     (phi[14])'
JACR39[1] -> (-(ADH65[3, 1] ADH69[2, 1]) + ADH65[2, 1] ADH69[3, 1])
>     (delta[1])' + (-(ADH62[3, 1] ADH69[2, 1]) +
>       ADH62[2, 1] ADH69[3, 1]) (delta[3])' -
>   (-(ADH61[3, 1] ADH69[2, 1]) + ADH61[2, 1] ADH69[3, 1]) (delta[4])' -
>   (-(ADH58[3, 2] ADH69[2, 1]) + ADH57[2, 2] ADH69[3, 1]) (delta[5])' -
>   (ADH56[2, 1] ADH69[3, 1] - ADH69[2, 1] Sin[phi[14]]) (phi[13])' -
>   ADH69[3, 1] Cos[phi[15]] (phi[14])' + ADH69[2, 1] (phi[15])'
JACR39[2] -> (ADH65[3, 1] ADH69[1, 1] - ADH65[1, 1] ADH69[3, 1])
>     (delta[1])' + (ADH62[3, 1] ADH69[1, 1] - ADH62[1, 1] ADH69[3, 1])
>     (delta[3])' - (ADH61[3, 1] ADH69[1, 1] - ADH61[1, 1] ADH69[3, 1])
>     (delta[4])' - (ADH58[3, 2] ADH69[1, 1] - ADH57[1, 2] ADH69[3, 1])
>     (delta[5])' - (-(ADH56[1, 1] ADH69[3, 1]) +
>       ADH69[1, 1] Sin[phi[14]]) (phi[13])' +
>   ADH69[3, 1] Sin[phi[15]] (phi[14])' - ADH69[1, 1] (phi[15])'
JACR39[3] -> (-(ADH65[2, 1] ADH69[1, 1]) + ADH65[1, 1] ADH69[2, 1])
>     (delta[1])' + (-(ADH62[2, 1] ADH69[1, 1]) +
>       ADH62[1, 1] ADH69[2, 1]) (delta[3])' -
>   (-(ADH61[2, 1] ADH69[1, 1]) + ADH61[1, 1] ADH69[2, 1]) (delta[4])' -
>   (-(ADH57[2, 2] ADH69[1, 1]) + ADH57[1, 2] ADH69[2, 1]) (delta[5])' -
>   (-(ADH56[2, 1] ADH69[1, 1]) + ADH56[1, 1] ADH69[2, 1]) (phi[13])' -
>   (-(ADH69[1, 1] Cos[phi[15]]) + ADH69[2, 1] Sin[phi[15]]) (phi[14])'
JACR39[4] -> (-(ADH69[3, 1] JACR29[5]) + ADH69[2, 1] JACR29[6])
>     (beta[9])' + (-(ADH65[3, 1] JACR29[5]) + ADH65[2, 1] JACR29[6])
>     (delta[1])' + (-(ADH62[3, 1] JACR29[5]) + ADH62[2, 1] JACR29[6])
>     (delta[3])' - (-(ADH61[3, 1] JACR29[5]) + ADH61[2, 1] JACR29[6])
>     (delta[4])' - (-(ADH58[3, 2] JACR29[5]) + ADH57[2, 2] JACR29[6])
>     (delta[5])' - (ADH56[2, 1] JACR29[6] - JACR29[5] Sin[phi[14]])
>     (phi[13])' - Cos[phi[15]] JACR29[6] (phi[14])' +
>   JACR29[5] (phi[15])'
JACR39[5] -> (ADH69[3, 1] JACR29[4] - ADH69[1, 1] JACR29[6]) (beta[9])' +
>   (ADH65[3, 1] JACR29[4] - ADH65[1, 1] JACR29[6]) (delta[1])' +
>   (ADH62[3, 1] JACR29[4] - ADH62[1, 1] JACR29[6]) (delta[3])' -
>   (ADH61[3, 1] JACR29[4] - ADH61[1, 1] JACR29[6]) (delta[4])' -
>   (ADH58[3, 2] JACR29[4] - ADH57[1, 2] JACR29[6]) (delta[5])' -
>   (-(ADH56[1, 1] JACR29[6]) + JACR29[4] Sin[phi[14]]) (phi[13])' +
>   JACR29[6] Sin[phi[15]] (phi[14])' - JACR29[4] (phi[15])'
```

```
JACR39[6] -> (-(ADH69[2, 1] JACR29[4]) + ADH69[1, 1] JACR29[5])
>     (beta[9])'' + (-(ADH65[2, 1] JACR29[4]) + ADH65[1, 1] JACR29[5])
>     (delta[1])'' + (-(ADH62[2, 1] JACR29[4]) + ADH62[1, 1] JACR29[5])
>     (delta[3])'' - (-(ADH61[2, 1] JACR29[4]) + ADH61[1, 1] JACR29[5])
>     (delta[4])'' - (-(ADH57[2, 2] JACR29[4]) + ADH57[1, 2] JACR29[5])
>     (delta[5])'' - (-(ADH56[2, 1] JACR29[4]) + ADH56[1, 1] JACR29[5])
>     (phi[13])'' - (-(Cos[phi[15]] JACR29[4]) + JACR29[5] Sin[phi[15]])
>     (phi[14])''
JACR31[6] -> -(JACR30[6] (beta[10]))'')
JACR40[1] -> JACR39[1] (beta[9])'' + JACR38[1] (delta[1])'' +
>    JACR37[1] (delta[3])'' - JACR36[1] (delta[4])'' -
>    JACR35[1] (delta[5])'' - JACR34[1] (phi[13])'' - JACR33[1] (phi[14])''
JACR40[2] -> JACR39[2] (beta[9])'' + JACR38[2] (delta[1])'' +
>    JACR37[2] (delta[3])'' - JACR36[2] (delta[4])'' -
>    JACR35[2] (delta[5])'' - JACR34[2] (phi[13])'' - JACR33[2] (phi[14])''
JACR40[3] -> JACR39[3] (beta[9])'' + JACR38[3] (delta[1])'' +
>    JACR37[3] (delta[3])'' - JACR36[3] (delta[4])'' -
>    JACR35[3] (delta[5])'' - JACR34[3] (phi[13])''
JACR40[4] -> JACR39[4] (beta[9])'' + JACR38[4] (delta[1])'' +
>    JACR37[4] (delta[3])'' - JACR36[4] (delta[4])'' -
>    JACR35[4] (delta[5])'' - JACR34[4] (phi[13])'' -
>    JACR33[4] (phi[14])'' - JACR32[4] (phi[15])''
JACR40[5] -> JACR39[5] (beta[9])'' + JACR38[5] (delta[1])'' +
>    JACR37[5] (delta[3])'' - JACR36[5] (delta[4])'' -
>    JACR35[5] (delta[5])'' - JACR34[5] (phi[13])'' -
>    JACR33[5] (phi[14])'' - JACR32[5] (phi[15])''
JACR40[6] -> JACR39[6] (beta[9])'' + JACR38[6] (delta[1])'' +
>    JACR37[6] (delta[3])'' - JACR36[6] (delta[4])'' -
>    JACR35[6] (delta[5])'' - JACR34[6] (phi[13])'' - JACR33[6] (phi[14])''
(beta[9])''' ->
>   (-(JACR28[5] JACR31[6]) - JACR28[6] JACR40[5] + JACR28[5] JACR40[6] +
>      JACR27[6] JACR28[5] (delta[3])''' -
>      JACR27[5] JACR28[6] (delta[3])''' -
>      JACR26[6] JACR28[5] (delta[4])''' +
>      JACR26[5] JACR28[6] (delta[4])''' -
>      JACR25[6] JACR28[5] (delta[5])''' + JACR25[5] JACR28[6] (delta[5])'''
>      ) / (JACR28[6] JACR29[5] - JACR28[5] JACR29[6])
(delta[1])''' ->
>   (-JACR40[5] - JACR29[5] (beta[9])''' - JACR27[5] (delta[3])''' +
>      JACR26[5] (delta[4])''' + JACR25[5] (delta[5])''') / JACR28[5]
(beta[10])''' ->
>   (-JACR40[4] - JACR29[4] (beta[9])''' - JACR28[4] (delta[1])''' -
>      JACR27[4] (delta[3])''' + JACR26[4] (delta[4])''' +
>      JACR25[4] (delta[5])''') / d
(phi[13])''' ->
>   (Cos[phi[15]] JACR40[1] - JACR40[2] Sin[phi[15]] +
>      ADH69[1, 1] Cos[phi[15]] (beta[9])''' -
>      ADH69[2, 1] Sin[phi[15]] (beta[9])''' - Sin[phi[15]] (beta[10])''' +
>      ADH65[1, 1] Cos[phi[15]] (delta[1])''' -
>      ADH65[2, 1] Sin[phi[15]] (delta[1])''' +
>      ADH62[1, 1] Cos[phi[15]] (delta[3])''' -
>      ADH62[2, 1] Sin[phi[15]] (delta[3])''' -
>      ADH61[1, 1] Cos[phi[15]] (delta[4])''' +
>      ADH61[2, 1] Sin[phi[15]] (delta[4])''' -
>      ADH57[1, 2] Cos[phi[15]] (delta[5])''' +
>      ADH57[2, 2] Sin[phi[15]] (delta[5])''') /
>     (ADH56[1, 1] Cos[phi[15]] - ADH56[2, 1] Sin[phi[15]])
(phi[14])''' ->
>   -(Csc[phi[15]] (-JACR40[1] - ADH69[1, 1] (beta[9])''' -
>        ADH65[1, 1] (delta[1])''' - ADH62[1, 1] (delta[3])''' +
>        ADH61[1, 1] (delta[4])''' + ADH57[1, 2] (delta[5])''' +
>        ADH56[1, 1] (phi[13])'''))
(phi[15])''' ->
>   JACR40[3] + ADH69[3, 1] (beta[9])'' + ADH65[3, 1] (delta[1])'' +
>    ADH62[3, 1] (delta[3])'' - ADH61[3, 1] (delta[4])'' -
>    ADH58[3, 2] (delta[5])'' - Sin[phi[14]] (phi[13])''
beta[6] -> 0
ADH73[2, 4] -> (e + h2) Cos[delta[3]]
ADH74[3, 4] -> -a2
                    -14 + ADH73[2, 4]
beta[7] -> -ArcSin[-----------------] - delta[3]
                     ADH74[3, 4]
ADH78[2, 4] -> -(ADH74[3, 4] Sin[beta[7]])
ADH77[2, 4] -> e + h2 + ADH78[2, 4]
ADH78[3, 4] -> ADH74[3, 4] Cos[beta[7]]
ADH76[2, 4] -> ADH77[2, 4] Cos[delta[3]] - ADH78[3, 4] Sin[delta[3]]
ADH76[3, 4] -> ADH78[3, 4] Cos[delta[3]] + ADH77[2, 4] Sin[delta[3]]
                                2
s[4] -> h3 - (-2 13 + Sqrt[4 13  -

                2     2              2              2
>         4 (13  + 14  - ADH76[2, 4]  - ADH76[3, 4] )]) / 2
phi[12] -> -3.14159
ADH83[1, 1] -> Cos[phi[12]]
phi[11] -> -ArcTan[ADH83[1, 1], 0]
ADH86[3, 2] -> -(Cos[delta[3]] Sin[beta[7]]) - Cos[beta[7]] Sin[delta[3]]
ADH85[3, 2] -> ADH86[3, 2]
ADH86[2, 2] -> Cos[beta[7]] Cos[delta[3]] - Sin[beta[7]] Sin[delta[3]]
ADH84[2, 2] -> ADH86[2, 2] Cos[phi[12]]
ADH84[3, 2] -> ADH85[3, 2] Cos[phi[12]]
phi[10] -> ArcTan[ADH84[2, 2], ADH84[3, 2]]
ADH73[3, 4] -> (e + h2) Sin[delta[3]]
ADH93[1, 4] -> 14 Sin[phi[12]]
ADH92[1, 2] -> -(Cos[phi[11]] Sin[phi[12]])
ADH92[1, 4] -> ADH93[1, 4] Cos[phi[11]] - 13 Sin[phi[11]]
ADH92[3, 2] -> Sin[phi[11]] Sin[phi[12]]
ADH92[3, 4] -> -(13 Cos[phi[11]]) - ADH93[1, 4] Sin[phi[11]]
ADH93[2, 4] -> -(14 Cos[phi[12]])
ADH97[3, 4] -> -h3 + ADH73[3, 4] + s[4]
ADH92[3, 1] -> -(Cos[phi[12]] Sin[phi[11]])
ADH95[1, 4] -> ADH92[1, 4] + (-h3 + s[4]) Sin[phi[11]]
ADH95[2, 4] ->
>   ADH93[2, 4] Cos[phi[10]] - ADH92[3, 4] Sin[phi[10]] -
>    Cos[phi[11]] (-h3 + s[4]) Sin[phi[10]]
ADH95[3, 4] ->
>   ADH92[3, 4] Cos[phi[10]] + Cos[phi[10]] Cos[phi[11]] (-h3 + s[4]) +
>    ADH93[2, 4] Sin[phi[10]]
ADH96[1, 4] ->
>   ADH73[2, 4] ADH92[1, 2] + ADH92[1, 4] + ADH97[3, 4] Sin[phi[11]]
ADH96[2, 4] ->
>   ADH93[2, 4] Cos[phi[10]] - ADH92[3, 4] Sin[phi[10]] -
>    ADH97[3, 4] Cos[phi[11]] Sin[phi[10]] +
>    ADH73[2, 4] (Cos[phi[10]] Cos[phi[12]] - ADH92[3, 2] Sin[phi[10]])
ADH96[3, 4] ->
>   ADH92[3, 4] Cos[phi[10]] + ADH97[3, 4] Cos[phi[10]] Cos[phi[11]] +
>    ADH93[2, 4] Sin[phi[10]] +
>    ADH73[2, 4] (ADH92[3, 2] Cos[phi[10]] + Cos[phi[12]] Sin[phi[10]])
ADH90[2, 3] -> -(Cos[phi[11]] Sin[phi[10]])
ADH90[3, 3] -> Cos[phi[10]] Cos[phi[11]]
ADH91[2, 3] -> -(Cos[phi[11]] Sin[phi[10]])
ADH91[3, 3] -> Cos[phi[10]] Cos[phi[11]]
ADH92[1, 1] -> Cos[phi[11]] Cos[phi[12]]
ADH95[2, 1] -> -(ADH92[3, 1] Sin[phi[10]]) + Cos[phi[10]] Sin[phi[12]]
ADH95[3, 1] -> ADH92[3, 1] Cos[phi[10]] + Sin[phi[10]] Sin[phi[12]]
ADH96[2, 1] -> -(ADH92[3, 1] Sin[phi[10]]) + Cos[phi[10]] Sin[phi[12]]
ADH96[3, 1] -> ADH92[3, 1] Cos[phi[10]] + Sin[phi[10]] Sin[phi[12]]
JACR46[4] -> ADH95[2, 4] ADH95[3, 1] - ADH95[2, 1] ADH95[3, 4]
JACR46[5] -> -(ADH95[1, 4] ADH95[3, 1]) + ADH92[1, 1] ADH95[3, 4]
JACR46[6] -> ADH95[1, 4] ADH95[2, 1] - ADH92[1, 1] ADH95[2, 4]
JACR47[4] -> ADH96[2, 4] ADH96[3, 1] - ADH96[2, 1] ADH96[3, 4]
JACR47[5] -> -(ADH96[1, 4] ADH96[3, 1]) + ADH92[1, 1] ADH96[3, 4]
JACR47[6] -> ADH96[1, 4] ADH96[2, 1] - ADH92[1, 1] ADH96[2, 4]
(beta[7])' -> ((-(ADH91[3, 3] JACR46[5]) + ADH91[2, 3] JACR46[6])
>      (delta[3])') / (ADH91[3, 3] JACR47[5] - ADH91[2, 3] JACR47[6])
(s[4])' -> (-(JACR47[5] (beta[7])') - JACR46[5] (delta[3])') / ADH91[2, 3]
(beta[8])' -> -((-(JACR47[4] (beta[7])') - JACR46[4] (delta[3])' -
>       Sin[phi[11]] (s[4])') / a2)
(phi[11])' -> (-(ADH90[3, 3] ADH96[2, 1] (beta[7])') +
>      ADH90[2, 3] ADH96[3, 1] (beta[7])' -
>      ADH90[3, 3] ADH95[2, 1] (delta[3])' +
>      ADH90[2, 3] ADH95[3, 1] (delta[3])') /
>    (ADH90[3, 3] Cos[phi[10]] - ADH90[2, 3] Sin[phi[10]])
(phi[12])' -> (-(ADH96[2, 1] (beta[7])') - ADH95[2, 1] (delta[3])' -
>      Cos[phi[10]] (phi[11])') / ADH90[2, 3]
(phi[10])' -> -(ADH92[1, 1] (beta[7])') - ADH92[1, 1] (delta[3])' -
>    Sin[phi[11]] (phi[12])'
JACR50[5] -> -(JACR47[6] (beta[7])') - JACR46[6] (delta[3])' -
>    ADH91[3, 3] (s[4])'
JACR50[6] -> JACR47[5] (beta[7])' + JACR46[5] (delta[3])' +
>    ADH91[2, 3] (s[4])'
JACR51[2] -> -(Sin[phi[10]] (phi[10])')
JACR51[3] -> Cos[phi[10]] (phi[10])'
JACR51[4] -> (Cos[phi[10]] JACR47[6] - JACR47[5] Sin[phi[10]])
>     (beta[7])' + (Cos[phi[10]] JACR46[6] - JACR46[5] Sin[phi[10]])
>     (delta[3])' + (ADH91[3, 3] Cos[phi[10]] - ADH91[2, 3] Sin[phi[10]])
>     (s[4])'
JACR51[5] -> JACR47[4] Sin[phi[10]] (beta[7])' +
>    JACR46[4] Sin[phi[10]] (delta[3])' +
>    Sin[phi[10]] Sin[phi[11]] (s[4])'
JACR51[6] -> -(Cos[phi[10]] JACR47[4] (beta[7])') -
>    Cos[phi[10]] JACR46[4] (delta[3])' -
>    Cos[phi[10]] Sin[phi[11]] (s[4])'
JACR52[1] -> (ADH90[3, 3] Cos[phi[10]] - ADH90[2, 3] Sin[phi[10]])
>    (phi[11])'
JACR52[2] -> -(ADH90[3, 3] (phi[10])') +
>    Sin[phi[10]] Sin[phi[11]] (phi[11])'
JACR52[3] -> ADH90[2, 3] (phi[10])' - Cos[phi[10]] Sin[phi[11]] (phi[11])'
JACR52[4] -> (-(ADH90[3, 3] JACR47[5]) + ADH90[2, 3] JACR47[6])
>     (beta[7])' + (-(ADH90[3, 3] JACR46[5]) + ADH90[2, 3] JACR46[6])
>     (delta[3])' + (-(ADH90[3, 3] ADH91[2, 3]) +
>       ADH90[2, 3] ADH91[3, 3]) (s[4])'
JACR52[5] -> (ADH90[3, 3] JACR47[4] - JACR47[6] Sin[phi[11]])
```

```
>     (beta[7])'' + (ADH90[3, 3] JACR46[4] - JACR46[6] Sin[phi[11]])
>     (delta[3])'' + (ADH90[3, 3] Sin[phi[11]] - ADH91[3, 3] Sin[phi[11]])
>     (s[4])'
JACR52[6] -> (-(ADH90[2, 3] JACR47[4]) + JACR47[5] Sin[phi[11]])
>     (beta[7])'' + (-(ADH90[2, 3] JACR46[4]) + JACR46[5] Sin[phi[11]])
>     (delta[3])'' + (-(ADH90[2, 3] Sin[phi[11]]) +
>       ADH91[2, 3] Sin[phi[11]]) (s[4])'
JACR53[4] -> (ADH91[3, 3] Cos[phi[10]] - ADH91[2, 3] Sin[phi[10]])
>     (phi[11])'' + (-(ADH90[3, 3] ADH91[2, 3]) + ADH90[2, 3] ADH91[3, 3])
>     (phi[12])'
JACR53[5] -> -(ADH91[3, 3] (phi[10])'') +
>    Sin[phi[10]] Sin[phi[11]] (phi[11])'' +
>    (ADH90[3, 3] Sin[phi[11]] - ADH91[3, 3] Sin[phi[11]]) (phi[12])'
JACR53[6] -> ADH91[2, 3] (phi[10])'' -
>    Cos[phi[10]] Sin[phi[11]] (phi[11])'' +
>    (-(ADH90[2, 3] Sin[phi[11]]) + ADH91[2, 3] Sin[phi[11]]) (phi[12])'
JACR54[1] -> (ADH95[3, 1] Cos[phi[10]] - ADH95[2, 1] Sin[phi[10]])
>     (phi[11])'' + (-(ADH90[3, 3] ADH95[2, 1]) + ADH90[2, 3] ADH95[3, 1])
>     (phi[12])''
JACR54[2] -> -(ADH95[3, 1] (phi[10])'') +
>    ADH92[1, 1] Sin[phi[10]] (phi[11])'' +
>    (ADH90[3, 3] ADH92[1, 1] - ADH95[3, 1] Sin[phi[11]]) (phi[12])'
JACR54[3] -> ADH95[2, 1] (phi[10])'' -
>    ADH92[1, 1] Cos[phi[10]] (phi[11])'' +
>    (-(ADH90[2, 3] ADH92[1, 1]) + ADH95[2, 1] Sin[phi[11]]) (phi[12])''
JACR54[4] -> (-(ADH95[3, 1] JACR47[5]) + ADH95[2, 1] JACR47[6])
>     (beta[7])'' + (-(ADH95[3, 1] JACR46[5]) + ADH95[2, 1] JACR46[6])
>     (delta[3])'' + (Cos[phi[10]] JACR46[6] - JACR46[5] Sin[phi[10]])
>     (phi[11])'' + (-(ADH90[3, 3] JACR46[5]) + ADH90[2, 3] JACR46[6])
>     (phi[12])''
JACR54[5] -> (ADH95[3, 1] JACR47[4] - ADH92[1, 1] JACR47[6]) (beta[7])'' +
>    (ADH95[3, 1] JACR46[4] - ADH92[1, 1] JACR46[6]) (delta[3])'' -
>    JACR46[6] (phi[10])'' + JACR46[4] Sin[phi[10]] (phi[11])'' +
>    (ADH90[3, 3] JACR46[4] - JACR46[6] Sin[phi[11]]) (phi[12])''
JACR54[6] -> (-(ADH95[2, 1] JACR47[4]) + ADH92[1, 1] JACR47[5])
>     (beta[7])'' + (-(ADH95[2, 1] JACR46[4]) + ADH92[1, 1] JACR46[5])
>     (delta[3])'' + JACR46[5] (phi[10])'' -
>    Cos[phi[10]] JACR46[4] (phi[11])'' +
>    (-(ADH90[2, 3] JACR46[4]) + JACR46[5] Sin[phi[11]]) (phi[12])''
JACR55[1] -> (-(ADH95[3, 1] ADH96[2, 1]) + ADH95[2, 1] ADH96[3, 1])
>     (delta[3])'' + (ADH96[3, 1] Cos[phi[10]] - ADH96[2, 1] Sin[phi[10]])
>     (phi[11])'' + (-(ADH90[3, 3] ADH96[2, 1]) + ADH90[2, 3] ADH96[3, 1])
>     (phi[12])''
JACR55[2] -> (ADH92[1, 1] ADH95[3, 1] - ADH92[1, 1] ADH96[3, 1])
>     (delta[3])'' - ADH96[3, 1] (phi[10])'' +
>    ADH92[1, 1] Sin[phi[10]] (phi[11])'' +
>    (ADH90[3, 3] ADH92[1, 1] - ADH96[3, 1] Sin[phi[11]]) (phi[12])''
JACR55[3] -> (-(ADH92[1, 1] ADH95[2, 1]) + ADH92[1, 1] ADH96[2, 1])
>     (delta[3])'' + ADH96[2, 1] (phi[10])'' -
>    ADH92[1, 1] Cos[phi[10]] (phi[11])'' +
>    (-(ADH90[2, 3] ADH92[1, 1]) + ADH96[2, 1] Sin[phi[11]]) (phi[12])''
JACR55[4] -> (-(ADH96[3, 1] JACR47[5]) + ADH96[2, 1] JACR47[6])
>     (beta[7])'' + (-(ADH95[3, 1] JACR47[5]) + ADH95[2, 1] JACR47[6])
>     (delta[3])'' + (Cos[phi[10]] JACR47[6] - JACR47[5] Sin[phi[10]])
>     (phi[11])'' + (-(ADH90[3, 3] JACR47[5]) + ADH90[2, 3] JACR47[6])
>     (phi[12])''
JACR55[5] -> (ADH96[3, 1] JACR47[4] - ADH92[1, 1] JACR47[6]) (beta[7])'' +
>    (ADH95[3, 1] JACR47[4] - ADH92[1, 1] JACR47[6]) (delta[3])'' -
>    JACR47[6] (phi[10])'' + JACR47[4] Sin[phi[10]] (phi[11])'' +
>    (ADH90[3, 3] JACR47[4] - JACR47[6] Sin[phi[11]]) (phi[12])''
JACR55[6] -> (-(ADH96[2, 1] JACR47[4]) + ADH92[1, 1] JACR47[5])
>     (beta[7])'' + (-(ADH95[2, 1] JACR47[4]) + ADH92[1, 1] JACR47[5])
>     (delta[3])'' + JACR47[5] (phi[10])'' -
>    Cos[phi[10]] JACR47[4] (phi[11])'' +
>    (-(ADH90[2, 3] JACR47[4]) + JACR47[5] Sin[phi[11]]) (phi[12])''
JACR56[1] -> JACR55[1] (beta[7])'' + JACR54[1] (delta[3])'' +
>    JACR52[1] (phi[12])''
JACR56[2] -> JACR55[2] (beta[7])'' + JACR54[2] (delta[3])'' +
>    JACR51[2] (phi[11])'' + JACR52[2] (phi[12])''
JACR56[3] -> JACR55[3] (beta[7])'' + JACR54[3] (delta[3])'' +
>    JACR51[3] (phi[11])'' + JACR52[3] (phi[12])''
JACR56[4] -> JACR55[4] (beta[7])'' + JACR54[4] (delta[3])'' +
>    JACR51[4] (phi[11])'' + JACR52[4] (phi[12])'' + JACR53[4] (s[4])'
JACR56[5] -> JACR55[5] (beta[7])'' + JACR54[5] (delta[3])'' +
>    JACR50[5] (phi[10])'' + JACR51[5] (phi[11])'' + JACR52[5] (phi[12])'' +
>    JACR53[5] (s[4])'
JACR56[6] -> JACR55[6] (beta[7])'' + JACR54[6] (delta[3])'' +
>    JACR50[6] (phi[10])'' + JACR51[6] (phi[11])'' + JACR52[6] (phi[12])'' +
>    JACR53[6] (s[4])''
(beta[7])''' ->
>   (-(ADH91[3, 3] JACR56[5]) + ADH91[2, 3] JACR56[6] -
>      ADH91[3, 3] JACR46[5] (delta[3])'' +
>      ADH91[2, 3] JACR46[6] (delta[3])'') /
>    (ADH91[3, 3] JACR47[5] - ADH91[2, 3] JACR47[6])
(s[4])''' -> (-JACR56[5] - JACR47[5] (beta[7])'' -
>      JACR46[5] (delta[3])'') / ADH91[2, 3]
(beta[8])''' ->
>   -((-JACR56[4] - JACR47[4] (beta[7])'' - JACR46[4] (delta[3])'' -
>       Sin[phi[11]] (s[4])''') / a2)
(phi[11])''' ->
>   (-(ADH90[3, 3] JACR56[2]) + ADH90[2, 3] JACR56[3] -
>      ADH90[3, 3] ADH96[2, 1] (beta[7])'' +
>      ADH90[2, 3] ADH96[3, 1] (beta[7])'' -
>      ADH90[3, 3] ADH95[2, 1] (delta[3])'' +
>      ADH90[2, 3] ADH95[3, 1] (delta[3])''') /
>    (ADH90[3, 3] Cos[phi[10]] - ADH90[2, 3] Sin[phi[10]])
(phi[12])''' ->
>   (-JACR56[2] - ADH96[2, 1] (beta[7])'' - ADH95[2, 1] (delta[3])'' -
>      Cos[phi[10]] (phi[11])''') / ADH90[2, 3]
(phi[10])''' ->
>   -JACR56[1] - ADH92[1, 1] (beta[7])'' - ADH92[1, 1] (delta[3])'' -
>    Sin[phi[11]] (phi[12])'''
ADH98[1, 4] -> -g + l1
                ADH98[1, 4]
beta[6] -> ArcSin[-----------]
                    a1
ADH104[2, 4] -> b1 Cos[delta[2]]
ADH102[2, 4] -> h2 + ADH104[2, 4]
ADH104[3, 4] -> b1 Sin[delta[2]]
ADH101[2, 4] -> ADH102[2, 4] Cos[delta[3]] - ADH104[3, 4] Sin[delta[3]]
ADH105[3, 4] -> a1 Cos[beta[6]]
ADH99[2, 4] -> -12 - l4 + ADH101[2, 4]
                ADH99[2, 4]
beta[5] -> -ArcSin[------------]
                ADH105[3, 4]
ADH101[3, 4] -> ADH104[3, 4] Cos[delta[3]] + ADH102[2, 4] Sin[delta[3]]
ADH100[3, 4] -> -h3 + ADH101[3, 4] + s[4]
ADH105[1, 4] -> a1 Sin[beta[6]]
ADH106[2, 4] -> -(ADH105[3, 4] Sin[beta[5]])
ADH106[3, 4] -> ADH105[3, 4] Cos[beta[5]]
                                                    2
s[3] -> (-2 ADH106[3, 4] + Sqrt[4 ADH106[3, 4]  -
                              2                2              2
>        4 (-ADH100[3, 4]  + ADH105[1, 4]  + ADH106[2, 4]  +
                            2              2              2
>          ADH106[3, 4]  - ADH98[1, 4]  - ADH99[2, 4] )]) / 2
ADH114[2, 1] -> Sin[beta[5]] Sin[beta[6]]
ADH114[3, 1] -> -(Cos[beta[5]] Sin[beta[6]])
ADH107[2, 1] ->
>   ADH114[3, 1] (Cos[delta[3]] Sin[delta[2]] +
>       Cos[delta[2]] Sin[delta[3]]) +
>    ADH114[2, 1] (Cos[delta[2]] Cos[delta[3]] -
>       Sin[delta[2]] Sin[delta[3]])
phi[9] -> -3.14159 - ArcTan[ADH107[2, 1] Sec[beta[6]]]
ADH107[3, 1] ->
>   ADH114[2, 1] (-(Cos[delta[3]] Sin[delta[2]]) -
>       Cos[delta[2]] Sin[delta[3]]) +
>    ADH114[3, 1] (Cos[delta[2]] Cos[delta[3]] -
>       Sin[delta[2]] Sin[delta[3]])
ADH115[1, 1] -> Cos[beta[6]] Cos[phi[9]] - ADH107[2, 1] Sin[phi[9]]
                        ADH115[1, 1]
phi[8] -> ArcTan[----------------------------,
                              2          2
                    ADH107[3, 1]  + ADH115[1, 1]
               ADH107[3, 1]
>     -------------------------------)
                   2              2
      ADH107[3, 1]  + ADH115[1, 1]
ADH107[2, 2] ->
>   Sin[beta[5]] (Cos[delta[3]] Sin[delta[2]] +
>       Cos[delta[2]] Sin[delta[3]]) +
>    Cos[beta[5]] (Cos[delta[2]] Cos[delta[3]] -
>       Sin[delta[2]] Sin[delta[3]])
ADH107[3, 2] ->
>   Cos[beta[5]] (-(Cos[delta[3]] Sin[delta[2]])) -
>       Cos[delta[2]] Sin[delta[3]]) +
>    Sin[beta[5]] (Cos[delta[2]] Cos[delta[3]] -
>       Sin[delta[2]] Sin[delta[3]])
ADH117[3, 2] -> Sin[phi[8]] Sin[phi[9]]
ADH116[2, 2] -> ADH107[2, 2] Cos[phi[9]]
ADH116[3, 2] -> ADH107[2, 2] ADH117[3, 2] + ADH107[3, 2] Cos[phi[8]]
phi[7] -> -ArcTan[ADH116[2, 2], ADH116[3, 2]]
ADH126[3, 4] -> h3 - s[4]
ADH127[2, 4] -> l2 + l4
ADH109[2, 4] -> -(h2 Cos[delta[2]])
ADH130[2, 4] ->
>   ADH127[2, 4] Cos[delta[3]] + ADH126[3, 4] Sin[delta[3]] +
>    s[3] Sin[delta[3]]
ADH130[3, 4] ->
>   ADH126[3, 4] Cos[delta[3]] + Cos[delta[3]] s[3] -
>    ADH127[2, 4] Sin[delta[3]]
ADH108[2, 4] -> -b1 + ADH109[2, 4]
```

```
ADH109[3, 4] -> h2 Sin[delta[2]]
ADH129[1, 4] -> g - 1i
ADH129[2, 4] ->
>   -bi + ADH109[2, 4] + ADH130[2, 4] Cos[delta[2]] +
>    ADH130[3, 4] Sin[delta[2]]
ADH129[3, 4] ->
>   ADH109[3, 4] + ADH130[3, 4] Cos[delta[2]] - ADH130[2, 4] Sin[delta[2]]
ADH117[1, 2] -> -(Cos[phi[8]] Sin[phi[9]])
ADH117[3, 1] -> -(Cos[phi[9]] Sin[phi[8]])
ADH119[2, 4] -> -(bi (Cos[phi[7]] Cos[phi[9]] - ADH117[3, 2] Sin[phi[7]]))
ADH119[3, 4] -> -(bi (ADH117[3, 2] Cos[phi[7]] + Cos[phi[9]] Sin[phi[7]]))
ADH122[2, 3] -> Cos[delta[3]] Sin[delta[2]] + Cos[delta[2]] Sin[delta[3]]
ADH122[3, 3] -> Cos[delta[2]] Cos[delta[3]] - Sin[delta[2]] Sin[delta[3]]
ADH124[2, 3] -> Cos[delta[3]] Sin[delta[2]] + Cos[delta[2]] Sin[delta[3]]
ADH124[3, 3] -> Cos[delta[2]] Cos[delta[3]] - Sin[delta[2]] Sin[delta[3]]
ADH119[1, 4] -> -(bi ADH117[1, 2])
ADH117[1, 1] -> Cos[phi[8]] Cos[phi[9]]
ADH120[1, 4] ->
>   g ADH117[1, 1] + ADH108[2, 4] ADH117[1, 2] + ADH109[3, 4] Sin[phi[8]]
ADH120[2, 4] ->
>   -(ADH109[3, 4] Cos[phi[8]] Sin[phi[7]]) +
>    ADH108[2, 4] (Cos[phi[7]] Cos[phi[9]] - ADH117[3, 2] Sin[phi[7]]) +
>    g (-(ADH117[3, 1] Sin[phi[7]]) + Cos[phi[7]] Sin[phi[9]])
ADH120[3, 4] ->
>   ADH109[3, 4] Cos[phi[7]] Cos[phi[8]] +
>    ADH108[2, 4] (ADH117[3, 2] Cos[phi[7]] + Cos[phi[9]] Sin[phi[7]]) +
>    g (ADH117[3, 1] Cos[phi[7]] + Sin[phi[7]] Sin[phi[9]])
ADH128[1, 4] ->
>   ADH117[1, 1] ADH129[1, 4] + ADH117[1, 2] ADH129[2, 4] +
>    ADH129[3, 4] Sin[phi[8]]
ADH128[2, 4] ->
>   -(ADH129[3, 4] Cos[phi[8]] Sin[phi[7]]) +
>    ADH129[2, 4] (Cos[phi[7]] Cos[phi[9]] - ADH117[3, 2] Sin[phi[7]]) +
>    ADH129[1, 4] (-(ADH117[3, 1] Sin[phi[7]]) + Cos[phi[7]] Sin[phi[9]])
ADH128[3, 4] ->
>   ADH129[3, 4] Cos[phi[7]] Cos[phi[8]] +
>    ADH129[2, 4] (ADH117[3, 2] Cos[phi[7]] + Cos[phi[9]] Sin[phi[7]]) +
>    ADH129[1, 4] (ADH117[3, 1] Cos[phi[7]] + Sin[phi[7]] Sin[phi[9]])
ADH118[2, 3] -> -(Cos[phi[8]] Sin[phi[7]])
ADH118[3, 3] -> Cos[phi[7]] Cos[phi[8]]
ADH119[2, 1] -> -(ADH117[3, 1] Sin[phi[7]]) + Cos[phi[7]] Sin[phi[9]]
ADH119[3, 1] -> ADH117[3, 1] Cos[phi[7]] + Sin[phi[7]] Sin[phi[9]]
ADH120[2, 1] -> -(ADH117[3, 1] Sin[phi[7]]) + Cos[phi[7]] Sin[phi[9]]
ADH120[3, 1] -> ADH117[3, 1] Cos[phi[7]] + Sin[phi[7]] Sin[phi[9]]
ADH121[1, 3] -> ADH117[1, 2] ADH122[2, 3] + ADH122[3, 3] Sin[phi[8]]
ADH121[2, 3] ->
>   -(ADH122[3, 3] Cos[phi[8]] Sin[phi[7]]) +
>    ADH122[2, 3] (Cos[phi[7]] Cos[phi[9]] - ADH117[3, 2] Sin[phi[7]])
ADH121[3, 3] ->
>   ADH122[3, 3] Cos[phi[7]] Cos[phi[8]] +
>    ADH122[2, 3] (ADH117[3, 2] Cos[phi[7]] + Cos[phi[9]] Sin[phi[7]])
ADH123[1, 3] -> ADH117[1, 2] ADH124[2, 3] + ADH124[3, 3] Sin[phi[8]]
ADH123[2, 3] ->
>   -(ADH124[3, 3] Cos[phi[8]] Sin[phi[7]]) +
>    ADH124[2, 3] (Cos[phi[7]] Cos[phi[9]] - ADH117[3, 2] Sin[phi[7]])
ADH123[3, 3] ->
>   ADH124[3, 3] Cos[phi[7]] Cos[phi[8]] +
>    ADH124[2, 3] (ADH117[3, 2] Cos[phi[7]] + Cos[phi[9]] Sin[phi[7]])
ADH128[2, 1] -> -(ADH117[3, 1] Sin[phi[7]]) + Cos[phi[7]] Sin[phi[9]]
ADH128[3, 1] -> ADH117[3, 1] Cos[phi[7]] + Sin[phi[7]] Sin[phi[9]]
JACR61[4] -> ADH119[2, 4] ADH119[3, 1] - ADH119[2, 1] ADH119[3, 4]
JACR61[5] -> -(ADH119[1, 4] ADH119[3, 1]) + ADH117[1, 1] ADH119[3, 4]
JACR61[6] -> ADH119[1, 4] ADH119[2, 1] - ADH117[1, 1] ADH119[2, 4]
JACR62[4] -> ADH120[2, 4] ADH120[3, 1] - ADH120[2, 1] ADH120[3, 4]
JACR62[5] -> -(ADH120[1, 4] ADH120[3, 1]) + ADH117[1, 1] ADH120[3, 4]
JACR62[6] -> ADH120[1, 4] ADH120[2, 1] - ADH117[1, 1] ADH120[2, 4]
JACR65[4] -> ADH128[2, 4] ADH128[3, 1] - ADH128[2, 1] ADH128[3, 4]
JACR65[5] -> -(ADH128[1, 4] ADH128[3, 1]) + ADH117[1, 1] ADH128[3, 4]
JACR65[6] -> ADH128[1, 4] ADH128[2, 1] - ADH117[1, 1] ADH128[2, 4]
(beta[5])' -> (ADH123[3, 3] JACR61[5] (delta[2]))' -
>      ADH123[2, 3] JACR61[6] (delta[2]))' +
>      ADH123[3, 3] JACR62[5] (delta[3]))' -
>      ADH123[2, 3] JACR62[6] (delta[3]))' -
>      ADH121[3, 3] ADH123[2, 3] (s[4]))' +
>      ADH121[2, 3] ADH123[3, 3] (s[4]))') /
>    (ADH123[3, 3] JACR65[5] - ADH123[2, 3] JACR65[6])
(s[3])' -> (-(JACR65[5] (beta[5]))') + JACR61[5] (delta[2]))' +
>      JACR62[5] (delta[3]))' + ADH121[2, 3] (s[4]))') / ADH123[2, 3]
(beta[6])' -> (-(JACR65[4] (beta[5]))') + JACR61[4] (delta[2]))' +
>      JACR62[4] (delta[3]))' - ADH123[1, 3] (s[3]))' + ADH121[1, 3] (s[4]))'
>      ) / a1
(phi[8])' -> (-(ADH118[3, 3] ADH128[2, 1] (beta[5]))') +
>      ADH118[2, 3] ADH128[3, 1] (beta[5]))' - ADH118[3, 3] (beta[6]))' +
>      ADH118[3, 3] ADH119[2, 1] (delta[2]))' -
>      ADH118[2, 3] ADH119[3, 1] (delta[2]))' +
>      ADH118[3, 3] ADH120[2, 1] (delta[3]))' -
>      ADH118[2, 3] ADH120[3, 1] (delta[3]))') /
>    (ADH118[3, 3] Cos[phi[7]] - ADH118[2, 3] Sin[phi[7]])
(phi[9])' -> (-(ADH128[2, 1] (beta[5]))') - (beta[6])' +
>      ADH119[2, 1] (delta[2]))' + ADH120[2, 1] (delta[3]))' -
>      Cos[phi[7]] (phi[8]))') / ADH118[2, 3]
(phi[7])' -> -(ADH117[1, 1] (beta[5]))') + ADH117[1, 1] (delta[2]))' +
>    ADH117[1, 1] (delta[3]))' - Sin[phi[8]] (phi[9]))'
JACR66[6] -> a1 (beta[6]))'
JACR65[5] -> -(JACR65[6] (beta[5]))') + JACR61[6] (delta[2]))' +
>    JACR62[6] (delta[3]))' - ADH123[3, 3] (s[3]))' + ADH121[3, 3] (s[4]))'
JACR65[6] -> JACR65[5] (beta[5]))' - JACR61[5] (delta[2]))' -
>    JACR62[5] (delta[3]))' + ADH123[2, 3] (s[3]))' - ADH121[2, 3] (s[4]))'
JACR69[2] -> -(Sin[phi[7]] (phi[7]))')
JACR69[3] -> Cos[phi[7]] (phi[7]))'
JACR69[4] -> (Cos[phi[7]] JACR65[6] - JACR65[5] Sin[phi[7]]) (beta[5]))' -
>    (Cos[phi[7]] JACR61[6] - JACR61[5] Sin[phi[7]]) (delta[2]))' -
>    (Cos[phi[7]] JACR62[6] - JACR62[5] Sin[phi[7]]) (delta[3]))' +
>    (ADH123[3, 3] Cos[phi[7]] - ADH123[2, 3] Sin[phi[7]]) (s[3]))' -
>    (ADH121[3, 3] Cos[phi[7]] - ADH121[2, 3] Sin[phi[7]]) (s[4]))'
JACR69[5] -> JACR65[4] Sin[phi[7]] (beta[5]))' -
>    JACR61[4] Sin[phi[7]] (delta[2]))' -
>    JACR62[4] Sin[phi[7]] (delta[3]))' +
>    ADH123[1, 3] Sin[phi[7]] (s[3]))' - ADH121[1, 3] Sin[phi[7]] (s[4]))'
JACR69[6] -> -(Cos[phi[7]] JACR65[4] (beta[5]))') +
>    Cos[phi[7]] JACR61[4] (delta[2]))' +
>    Cos[phi[7]] JACR62[4] (delta[3]))' -
>    ADH123[1, 3] Cos[phi[7]] (s[3]))' + ADH121[1, 3] Cos[phi[7]] (s[4]))'
JACR70[1] -> (ADH118[3, 3] Cos[phi[7]] - ADH118[2, 3] Sin[phi[7]])
>    (phi[8]))'
JACR70[2] -> -(ADH118[3, 3] (phi[7]))') + Sin[phi[7]] Sin[phi[8]] (phi[8]))'
JACR70[3] -> ADH118[2, 3] (phi[7]))' - Cos[phi[7]] Sin[phi[8]] (phi[8]))'
JACR70[4] -> (-(ADH118[3, 3] JACR65[5]) + ADH118[2, 3] JACR65[6])
>     (beta[5]))' - (-(ADH118[3, 3] JACR61[5]) + ADH118[2, 3] JACR61[6])
>     (delta[2]))' - (-(ADH118[3, 3] JACR62[5]) + ADH118[2, 3] JACR62[6])
>     (delta[3]))' + (-(ADH118[3, 3] ADH123[2, 3]) +
>       ADH118[2, 3] ADH123[3, 3]) (s[3]))' -
>    (-(ADH118[3, 3] ADH121[2, 3]) + ADH118[2, 3] ADH121[3, 3]) (s[4]))'
JACR70[5] -> (ADH118[3, 3] JACR65[4] - JACR65[6] Sin[phi[8]])
>     (beta[5]))' - (ADH118[3, 3] JACR61[4] - JACR61[6] Sin[phi[8]])
>     (delta[2]))' - (ADH118[3, 3] JACR62[4] - JACR62[6] Sin[phi[8]])
>     (delta[3]))' + (ADH118[3, 3] ADH123[1, 3] -
>       ADH123[3, 3] Sin[phi[8]]) (s[3]))' -
>    (ADH118[3, 3] ADH121[1, 3] - ADH121[3, 3] Sin[phi[8]]) (s[4]))'
JACR70[6] -> (-(ADH118[2, 3] JACR65[4]) + JACR65[5] Sin[phi[8]])
>     (beta[5]))' - (-(ADH118[2, 3] JACR61[4]) + JACR61[5] Sin[phi[8]])
>     (delta[2]))' - (-(ADH118[2, 3] JACR62[4]) + JACR62[5] Sin[phi[8]])
>     (delta[3]))' + (-(ADH118[2, 3] ADH123[1, 3]) +
>       ADH123[2, 3] Sin[phi[8]]) (s[3]))' -
>    (-(ADH118[2, 3] ADH121[1, 3]) + ADH121[2, 3] Sin[phi[8]]) (s[4]))'
JACR71[1] -> (ADH119[3, 1] Cos[phi[7]] - ADH119[2, 1] Sin[phi[7]])
>     (phi[8]))' + (-(ADH118[3, 3] ADH119[2, 1]) +
>       ADH118[2, 3] ADH119[3, 1]) (phi[9]))'
JACR71[2] -> -(ADH119[3, 1] (phi[7]))') +
>    ADH117[1, 1] Sin[phi[7]] (phi[8]))' +
>    (ADH117[1, 1] ADH118[3, 3] - ADH119[3, 1] Sin[phi[8]]) (phi[9]))'
JACR71[3] -> ADH119[2, 1] (phi[7]))' -
>    ADH117[1, 1] Cos[phi[7]] (phi[8]))' +
>    (-(ADH117[1, 1] ADH118[2, 3]) + ADH119[2, 1] Sin[phi[8]]) (phi[9]))'
JACR71[4] -> (-(ADH119[3, 1] JACR65[5]) + ADH119[2, 1] JACR65[6])
>     (beta[5]))' - (-(ADH119[3, 1] JACR61[5]) + ADH119[2, 1] JACR61[6])
>     (delta[2]))' - (-(ADH119[3, 1] JACR62[5]) + ADH119[2, 1] JACR62[6])
>     (delta[3]))' + (Cos[phi[7]] JACR61[6] - JACR61[5] Sin[phi[7]])
>     (phi[8]))' + (-(ADH118[3, 3] JACR61[5]) + ADH118[2, 3] JACR61[6])
>     (phi[9]))' + (-(ADH119[3, 1] ADH123[2, 3]) +
>       ADH119[2, 1] ADH123[3, 3]) (s[3]))' -
>    (-(ADH119[3, 1] ADH121[2, 3]) + ADH119[2, 1] ADH121[3, 3]) (s[4]))'
JACR71[5] -> (ADH119[3, 1] JACR65[4] - ADH117[1, 1] JACR65[6])
>     (beta[5]))' - (ADH119[3, 1] JACR61[4] - ADH117[1, 1] JACR61[6])
>     (delta[2]))' - (ADH119[3, 1] JACR62[4] - ADH117[1, 1] JACR62[6])
>     (delta[3]))' - JACR61[6] (phi[7]))' +
>    JACR61[4] Sin[phi[7]] (phi[8]))' +
>    (ADH118[3, 3] JACR61[4] - JACR61[6] Sin[phi[8]]) (phi[9]))' +
>    (ADH119[3, 1] ADH123[1, 3] - ADH117[1, 1] ADH123[3, 3]) (s[3]))' -
>    (ADH119[3, 1] ADH121[1, 3] - ADH117[1, 1] ADH121[3, 3]) (s[4]))'
JACR71[6] -> (-(ADH119[2, 1] JACR65[4]) + ADH117[1, 1] JACR65[5])
>     (beta[5]))' - (-(ADH119[2, 1] JACR61[4]) + ADH117[1, 1] JACR61[5])
>     (delta[2]))' - (-(ADH119[2, 1] JACR62[4]) + ADH117[1, 1] JACR62[5])
>     (delta[3]))' + JACR61[5] (phi[7]))' -
>    Cos[phi[7]] JACR61[4] (phi[8]))' +
>    (-(ADH118[2, 3] JACR61[4]) + JACR61[5] Sin[phi[8]]) (phi[9]))' +
>    (-(ADH119[2, 1] ADH123[1, 3]) + ADH117[1, 1] ADH123[2, 3]) (s[3]))' -
>    (-(ADH119[2, 1] ADH121[1, 3]) + ADH117[1, 1] ADH121[2, 3]) (s[4]))'
JACR72[1] -> -((-(ADH119[3, 1] ADH120[2, 1]) + ADH119[2, 1] ADH120[3, 1])
>       (delta[2]))') + (ADH120[3, 1] Cos[phi[7]] -
>       ADH120[2, 1] Sin[phi[7]]) (phi[8]))' +
>    (-(ADH118[3, 3] ADH120[2, 1]) + ADH118[2, 3] ADH120[3, 1]) (phi[9]))'
```

```
JACR72[2] -> -((ADH117[1, 1] ADH119[3, 1] - ADH117[1, 1] ADH120[3, 1])
>       (delta[2])') - ADH120[3, 1] (phi[7])' +
>    ADH117[1, 1] Sin[phi[7]] (phi[8])' +
>    (ADH117[1, 1] ADH118[3, 3] - ADH120[3, 1] Sin[phi[8]]) (phi[9])'
JACR72[3] -> -((-(ADH117[1, 1] ADH119[2, 1]) + ADH117[1, 1] ADH120[2, 1])
>       (delta[2])') + ADH120[2, 1] (phi[7])' -
>    ADH117[1, 1] Cos[phi[7]] (phi[8])' +
>    (-(ADH117[1, 1] ADH118[2, 3]) + ADH120[2, 1] Sin[phi[8]]) (phi[9])'
JACR72[4] -> (-(ADH120[3, 1] JACR65[5]) + ADH120[2, 1] JACR65[6])
>     (beta[5])' - (-(ADH119[3, 1] JACR62[5]) + ADH119[2, 1] JACR62[6])
>     (delta[2])' - (-(ADH120[3, 1] JACR62[5]) + ADH120[2, 1] JACR62[6])
>     (delta[3])' + (Cos[phi[7]] JACR62[6] - JACR62[5] Sin[phi[7]])
>     (phi[8])' + (-(ADH118[3, 3] JACR62[5]) + ADH118[2, 3] JACR62[6])
>     (phi[9])' + (-(ADH120[3, 1] ADH123[2, 3]) +
>       ADH120[2, 1] ADH123[3, 3]) (s[3])' -
>    (-(ADH120[3, 1] ADH121[2, 3]) + ADH120[2, 1] ADH121[3, 3]) (s[4])'
JACR72[5] -> (ADH120[3, 1] JACR65[4] - ADH117[1, 1] JACR65[6])
>     (beta[5])' - (ADH119[3, 1] JACR62[4] - ADH117[1, 1] JACR62[6])
>     (delta[2])' - (ADH120[3, 1] JACR62[4] - ADH117[1, 1] JACR62[6])
>     (delta[3])' - JACR62[6] (phi[7])' +
>    JACR62[4] Sin[phi[7]] (phi[8])' +
>    (ADH118[3, 3] JACR62[4] - JACR62[6] Sin[phi[8]]) (phi[9])' +
>    (ADH120[3, 1] ADH123[1, 3] - ADH117[1, 1] ADH123[3, 3]) (s[3])' -
>    (ADH120[3, 1] ADH121[1, 3] - ADH117[1, 1] ADH121[3, 3]) (s[4])'
JACR72[6] -> (-(ADH120[2, 1] JACR65[4]) + ADH117[1, 1] JACR65[5])
>     (beta[5])' - (-(ADH119[2, 1] JACR62[4]) + ADH117[1, 1] JACR62[5])
>     (delta[2])' - (-(ADH120[2, 1] JACR62[4]) + ADH117[1, 1] JACR62[5])
>     (delta[3])' + JACR62[5] (phi[7])' -
>    Cos[phi[7]] JACR62[4] (phi[8])' +
>    (-(ADH118[2, 3] JACR62[4]) + JACR62[5] Sin[phi[8]]) (phi[9])' +
>    (-(ADH120[2, 1] ADH123[1, 3]) + ADH117[1, 1] ADH123[2, 3]) (s[3])' -
>    (-(ADH120[2, 1] ADH121[1, 3]) + ADH117[1, 1] ADH121[2, 3]) (s[4])'
JACR73[4] -> -((-(ADH119[3, 1] ADH121[2, 3]) + ADH119[2, 1] ADH121[3, 3])
>       (delta[2])') - (-(ADH120[3, 1] ADH121[2, 3]) +
>       ADH120[2, 1] ADH121[3, 3]) (delta[3])' +
>    (ADH121[3, 3] Cos[phi[7]] - ADH121[2, 3] Sin[phi[7]]) (phi[8])' +
>    (-(ADH118[3, 3] ADH121[2, 3]) + ADH118[2, 3] ADH121[3, 3]) (phi[9])'
JACR73[5] -> -((ADH119[3, 1] ADH121[1, 3] - ADH117[1, 1] ADH121[3, 3])
>       (delta[2])') - (ADH120[3, 1] ADH121[1, 3] -
>       ADH117[1, 1] ADH121[3, 3]) (delta[3])' - ADH121[3, 3] (phi[7])' +
>    ADH121[1, 3] Sin[phi[7]] (phi[8])' +
>    (ADH118[3, 3] ADH121[1, 3] - ADH121[3, 3] Sin[phi[8]]) (phi[9])'
JACR73[6] -> -((-(ADH119[2, 1] ADH121[1, 3]) + ADH117[1, 1] ADH121[2, 3])
>       (delta[2])') - (-(ADH120[2, 1] ADH121[1, 3]) +
>       ADH117[1, 1] ADH121[2, 3]) (delta[3])' + ADH121[2, 3] (phi[7])' -
>    ADH121[1, 3] Cos[phi[7]] (phi[8])' +
>    (-(ADH118[2, 3] ADH121[1, 3]) + ADH121[2, 3] Sin[phi[8]]) (phi[9])'
JACR74[4] -> -((-(ADH119[3, 1] ADH123[2, 3]) + ADH119[2, 1] ADH123[3, 3])
>       (delta[2])') - (-(ADH120[3, 1] ADH123[2, 3]) +
>       ADH120[2, 1] ADH123[3, 3]) (delta[3])' +
>    (ADH123[3, 3] Cos[phi[7]] - ADH123[2, 3] Sin[phi[7]]) (phi[8])' +
>    (-(ADH118[3, 3] ADH123[2, 3]) + ADH118[2, 3] ADH123[3, 3]) (phi[9])'
JACR74[5] -> -((ADH119[3, 1] ADH123[1, 3] - ADH117[1, 1] ADH123[3, 3])
>       (delta[2])') - (ADH120[3, 1] ADH123[1, 3] -
>       ADH117[1, 1] ADH123[3, 3]) (delta[3])' - ADH123[3, 3] (phi[7])' +
>    ADH123[1, 3] Sin[phi[7]] (phi[8])' +
>    (ADH118[3, 3] ADH123[1, 3] - ADH123[3, 3] Sin[phi[8]]) (phi[9])'
JACR74[6] -> -((-(ADH119[2, 1] ADH123[1, 3]) + ADH117[1, 1] ADH123[2, 3])
>       (delta[2])') - (-(ADH120[2, 1] ADH123[1, 3]) +
>       ADH117[1, 1] ADH123[2, 3]) (delta[3])' + ADH123[2, 3] (phi[7])' -
>    ADH123[1, 3] Cos[phi[7]] (phi[8])' +
>    (-(ADH118[2, 3] ADH123[1, 3]) + ADH123[2, 3] Sin[phi[8]]) (phi[9])'
JACR75[1] -> -((-(ADH119[3, 1] ADH128[2, 1]) + ADH119[2, 1] ADH128[3, 1])
>       (delta[2])') - (-(ADH120[3, 1] ADH128[2, 1]) +
>       ADH120[2, 1] ADH128[3, 1]) (delta[3])' +
>    (ADH128[3, 1] Cos[phi[7]] - ADH128[2, 1] Sin[phi[7]]) (phi[8])' +
>    (-(ADH118[3, 3] ADH128[2, 1]) + ADH118[2, 3] ADH128[3, 1]) (phi[9])'
JACR75[2] -> -((ADH117[1, 1] ADH119[3, 1] - ADH117[1, 1] ADH128[3, 1])
>       (delta[2])') - (ADH117[1, 1] ADH120[3, 1] -
>       ADH117[1, 1] ADH128[3, 1]) (delta[3])' - ADH128[3, 1] (phi[7])' +
>    ADH117[1, 1] Sin[phi[7]] (phi[8])' +
>    (ADH117[1, 1] ADH118[3, 3] - ADH128[3, 1] Sin[phi[8]]) (phi[9])'
JACR75[3] -> -((-(ADH117[1, 1] ADH119[2, 1]) + ADH117[1, 1] ADH128[2, 1])
>       (delta[2])') - (-(ADH117[1, 1] ADH120[2, 1]) +
>       ADH117[1, 1] ADH128[2, 1]) (delta[3])' + ADH128[2, 1] (phi[7])' -
>    ADH117[1, 1] Cos[phi[7]] (phi[8])' +
>    (-(ADH117[1, 1] ADH118[2, 3]) + ADH128[2, 1] Sin[phi[8]]) (phi[9])'
JACR75[4] -> (-(ADH128[3, 1] JACR65[5]) + ADH128[2, 1] JACR65[6])
>     (beta[5])' - (-(ADH119[3, 1] JACR65[5]) + ADH119[2, 1] JACR65[6])
>     (delta[2])' - (-(ADH120[3, 1] JACR65[5]) + ADH120[2, 1] JACR65[6])
>     (delta[3])' + (Cos[phi[7]] JACR65[6] - JACR65[5] Sin[phi[7]])
>     (phi[8])' + (-(ADH118[3, 3] JACR65[5]) + ADH118[2, 3] JACR65[6])
>     (phi[9])'
JACR75[5] -> (ADH128[3, 1] JACR65[4] - ADH117[1, 1] JACR65[6])
>     (beta[5])' - (ADH119[3, 1] JACR65[4] - ADH117[1, 1] JACR65[6])
>     (delta[2])' - (ADH120[3, 1] JACR65[4] - ADH117[1, 1] JACR65[6])
>     (delta[3])' - JACR65[6] (phi[7])' +
>    JACR65[4] Sin[phi[7]] (phi[8])' +
>    (ADH118[3, 3] JACR65[4] - JACR65[6] Sin[phi[8]]) (phi[9])'
JACR75[6] -> (-(ADH128[2, 1] JACR65[4]) + ADH117[1, 1] JACR65[5])
>     (beta[5])' - (-(ADH119[2, 1] JACR65[4]) + ADH117[1, 1] JACR65[5])
>     (delta[2])' - (-(ADH120[2, 1] JACR65[4]) + ADH117[1, 1] JACR65[5])
>     (delta[3])' + JACR65[5] (phi[7])' -
>    Cos[phi[7]] JACR65[4] (phi[8])' +
>    (-(ADH118[2, 3] JACR65[4]) + JACR65[5] Sin[phi[8]]) (phi[9])'
JACR67[6] -> -(JACR66[6] (beta[6])')
JACR76[1] -> JACR75[1] (beta[5])' - JACR71[1] (delta[2])' -
>    JACR72[1] (delta[3])' + JACR70[1] (phi[9])'
JACR76[2] -> JACR75[2] (beta[5])' - JACR71[2] (delta[2])' -
>    JACR72[2] (delta[3])' + JACR69[2] (phi[8])' + JACR70[2] (phi[9])'
JACR76[3] -> JACR75[3] (beta[5])' - JACR71[3] (delta[2])' -
>    JACR72[3] (delta[3])' + JACR69[3] (phi[8])' + JACR70[3] (phi[9])'
JACR76[4] -> JACR75[4] (beta[5])' - JACR71[4] (delta[2])' -
>    JACR72[4] (delta[3])' + JACR69[4] (phi[8])' + JACR70[4] (phi[9])' +
>    JACR74[4] (s[3])' - JACR73[4] (s[4])'
JACR76[5] -> JACR75[5] (beta[5])' - JACR71[5] (delta[2])' -
>    JACR72[5] (delta[3])' + JACR68[5] (phi[7])' + JACR69[5] (phi[8])' +
>    JACR70[5] (phi[9])' + JACR74[5] (s[3])' - JACR73[5] (s[4])'
JACR76[6] -> JACR75[6] (beta[5])' - JACR71[6] (delta[2])' -
>    JACR72[6] (delta[3])' + JACR68[6] (phi[7])' + JACR69[6] (phi[8])' +
>    JACR70[6] (phi[9])' + JACR74[6] (s[3])' - JACR73[6] (s[4])'
(beta[5])'' ->
>   (-(ADH123[2, 3] JACR67[6]) - ADH123[3, 3] JACR76[5] +
>      ADH123[2, 3] JACR76[6] + ADH123[3, 3] JACR61[5] (delta[2])'' -
>      ADH123[2, 3] JACR61[6] (delta[2])'' +
>      ADH123[3, 3] JACR62[5] (delta[3])'' -
>      ADH123[2, 3] JACR62[6] (delta[3])'' -
>      ADH121[3, 3] ADH123[2, 3] (s[4])'' +
>      ADH121[2, 3] ADH123[3, 3] (s[4])'') /
>    (ADH123[3, 3] JACR65[5] - ADH123[2, 3] JACR65[6])
(s[3])'' -> (-JACR76[5] - JACR65[5] (beta[5])'' +
>      JACR61[5] (delta[2])'' + JACR62[5] (delta[3])'' +
>      ADH121[2, 3] (s[4])'') / ADH123[2, 3]
(beta[6])'' ->
>   (-JACR76[4] - JACR65[4] (beta[5])'' + JACR61[4] (delta[2])'' +
>      JACR62[4] (delta[3])'' - ADH123[1, 3] (s[3])'' +
>      ADH121[1, 3] (s[4])'') / ai
(phi[8])'' -> (-(ADH118[3, 3] JACR76[2]) + ADH118[2, 3] JACR76[3] -
>      ADH118[3, 3] ADH128[2, 1] (beta[5])'' +
>      ADH118[2, 3] ADH128[3, 1] (beta[5])'' - ADH118[3, 3] (beta[6])'' +
>      ADH118[3, 3] ADH119[2, 1] (delta[2])'' -
>      ADH118[2, 3] ADH119[3, 1] (delta[2])'' +
>      ADH118[3, 3] ADH120[2, 1] (delta[3])'' -
>      ADH118[2, 3] ADH120[3, 1] (delta[3])'') /
>    (ADH118[3, 3] Cos[phi[7]] - ADH118[2, 3] Sin[phi[7]])
(phi[9])'' -> (-JACR76[2] - ADH128[2, 1] (beta[5])'' - (beta[6])'' +
>      ADH119[2, 1] (delta[2])'' + ADH120[2, 1] (delta[3])'' -
>      Cos[phi[7]] (phi[8])'') / ADH118[2, 3]
(phi[7])'' -> -JACR76[1] - ADH117[1, 1] (beta[5])'' +
>    ADH117[1, 1] (delta[2])'' + ADH117[1, 1] (delta[3])'' -
>    Sin[phi[8]] (phi[9])''
ADH131[1, 4] -> g - 11
                ADH131[1, 4]
beta[2] -> ArcSin[------------]
                     ai
ADH133[2, 4] -> -12 - 14
ADH134[2, 4] -> bi Cos[delta[1]]
ADH134[3, 4] -> bi Sin[delta[1]]
ADH135[3, 4] -> ai Cos[beta[2]]
ADH136[2, 4] ->
>   ADH133[2, 4] + ADH67[2, 4] + ADH134[2, 4] Cos[delta[3]] -
>    ADH134[3, 4] Sin[delta[3]]
                  ADH136[2, 4]
beta[i] -> -ArcSin[------------]
                  ADH135[3, 4]
ADH133[3, 4] -> -h3 + s[4]
ADH131[2, 4] ->
>   ADH133[2, 4] + ADH67[2, 4] + ADH134[2, 4] Cos[delta[3]] -
>    ADH134[3, 4] Sin[delta[3]]
ADH131[3, 4] ->
>   ADH133[3, 4] + ADH67[3, 4] + ADH134[3, 4] Cos[delta[3]] +
>    ADH134[2, 4] Sin[delta[3]]
ADH135[1, 4] -> ai Sin[beta[2]]
ADH137[2, 4] -> -(ADH135[3, 4] Sin[beta[1]])
ADH137[3, 4] -> ADH135[3, 4] Cos[beta[1]]
                                                  2
s[1] -> (-2 ADH137[3, 4] + Sqrt[4 ADH137[3, 4]  -
                          2                2              2
>        4 (-ADH131[1, 4]  - ADH131[2, 4]  - ADH131[3, 4]  +
                         2                2              2
>          ADH135[1, 4]  + ADH137[2, 4]  + ADH137[3, 4] )]) / 2
ADH143[2, 1] -> Sin[beta[1]] Sin[beta[2]]
```

```
ADH143[3, i] -> -(Cos[beta[i]] Sin[beta[2]])
ADH139[2, i] -> ADH143[2, i] Cos[delta[3]] + ADH143[3, i] Sin[delta[3]]
ADH139[3, i] -> ADH143[3, i] Cos[delta[3]] - ADH143[2, i] Sin[delta[3]]
ADH138[2, i] -> ADH139[2, i] Cos[delta[i]] + ADH139[3, i] Sin[delta[i]]
phi[3] -> -3.14159 - ArcTan[ADH138[2, i] Sec[beta[2]]]
ADH144[i, i] ->
>   Cos[beta[2]] Cos[phi[3]] - ADH139[2, i] Cos[delta[i]] Sin[phi[3]] -
>    ADH139[3, i] Sin[delta[i]] Sin[phi[3]]
ADH144[3, i] -> ADH139[3, i] Cos[delta[i]] - ADH139[2, i] Sin[delta[i]]
                        ADH144[i, i]
phi[2] -> ArcTan[-----------------------------,
                                2                2
                    ADH144[i, i]  + ADH144[3, i]
                ADH144[3, i]
>    -------------------------------]
                  2                2
     ADH144[i, i]  + ADH144[3, i]
ADH139[2, 2] -> Cos[beta[i]] Cos[delta[3]] + Sin[beta[i]] Sin[delta[3]]
ADH139[3, 2] -> Cos[delta[3]] Sin[beta[i]] - Cos[beta[i]] Sin[delta[3]]
ADH144[i, 2] ->
>   -(ADH139[2, 2] Cos[delta[i]] Sin[phi[3]]) -
>    ADH139[3, 2] Sin[delta[i]] Sin[phi[3]]
ADH144[3, 2] -> ADH139[3, 2] Cos[delta[i]] - ADH139[2, 2] Sin[delta[i]]
ADH144[2, 2] ->
>   ADH139[2, 2] Cos[delta[i]] Cos[phi[3]] +
>    ADH139[3, 2] Cos[phi[3]] Sin[delta[i]]
ADH146[3, 2] -> ADH144[3, 2] Cos[phi[2]] - ADH144[i, 2] Sin[phi[2]]
phi[i] -> -ArcTan[ADH144[2, 2], ADH146[3, 2]]
ADH158[3, 4] -> h3 - s[4]
ADH159[2, 4] -> l2 + l4
ADH162[2, 4] ->
>   ADH159[2, 4] Cos[delta[3]] + ADH158[3, 4] Sin[delta[3]] +
>    s[i] Sin[delta[3]]
ADH162[3, 4] ->
>   ADH158[3, 4] Cos[delta[3]] + Cos[delta[3]] s[i] -
>    ADH159[2, 4] Sin[delta[3]]
ADH145[i, 4] -> bi Sin[phi[3]]
ADH145[2, 4] -> -(bi Cos[phi[3]])
ADH151[2, 4] -> -(h2 Cos[delta[i]])
ADH151[3, 4] -> h2 Sin[delta[i]]
ADH161[i, 4] -> -g + li
ADH161[2, 4] ->
>   -(h2 Cos[delta[i]]) + ADH162[2, 4] Cos[delta[i]] +
>    ADH162[3, 4] Sin[delta[i]]
ADH161[3, 4] ->
>   ADH162[3, 4] Cos[delta[i]] + h2 Sin[delta[i]] -
>    ADH162[2, 4] Sin[delta[i]]
ADH149[3, 4] -> -(ADH145[1, 4] Sin[phi[2]])
ADH148[2, 4] -> ADH145[2, 4] Cos[phi[i]] - ADH149[3, 4] Sin[phi[i]]
ADH148[3, 4] -> ADH149[3, 4] Cos[phi[i]] + ADH145[2, 4] Sin[phi[i]]
ADH149[i, 2] -> -(Cos[phi[2]] Sin[phi[3]])
ADH149[i, 4] -> ADH145[i, 4] Cos[phi[2]]
ADH149[3, i] -> -(Cos[phi[3]] Sin[phi[2]])
ADH149[3, 2] -> Sin[phi[2]] Sin[phi[3]]
ADH154[2, 3] -> Cos[delta[3]] Sin[delta[i]] + Cos[delta[i]] Sin[delta[3]]
ADH154[3, 3] -> Cos[delta[i]] Cos[delta[3]] - Sin[delta[i]] Sin[delta[3]]
ADH156[2, 3] -> Cos[delta[3]] Sin[delta[i]] + Cos[delta[i]] Sin[delta[3]]
ADH156[3, 3] -> Cos[delta[i]] Cos[delta[3]] - Sin[delta[i]] Sin[delta[3]]
ADH149[i, i] -> Cos[phi[2]] Cos[phi[3]]
ADH150[i, 4] ->
>   -(g ADH149[i, i]) + ADH149[i, 4] + ADH149[i, 2] ADH151[2, 4] +
>    ADH151[3, 4] Sin[phi[2]]
ADH150[2, 4] ->
>   ADH145[2, 4] Cos[phi[i]] - ADH149[3, 4] Sin[phi[i]] -
>    ADH151[3, 4] Cos[phi[2]] Sin[phi[i]] +
>    ADH151[2, 4] (Cos[phi[i]] Cos[phi[3]] - ADH149[3, 2] Sin[phi[i]]) -
>    g (-(ADH149[3, i] Sin[phi[i]]) + Cos[phi[i]] Sin[phi[3]])
ADH150[3, 4] ->
>   ADH149[3, 4] Cos[phi[i]] + ADH151[3, 4] Cos[phi[i]] Cos[phi[2]] +
>    ADH145[2, 4] Sin[phi[i]] +
>    ADH151[2, 4] (ADH149[3, 2] Cos[phi[i]] + Cos[phi[3]] Sin[phi[i]]) -
>    g (ADH149[3, i] Cos[phi[i]] + Sin[phi[i]] Sin[phi[3]])
ADH160[i, 4] ->
>   ADH149[i, 4] + ADH149[i, i] ADH161[i, 4] +
>    ADH149[i, 2] ADH161[2, 4] + ADH161[3, 4] Sin[phi[2]]
ADH160[2, 4] ->
>   ADH145[2, 4] Cos[phi[i]] - ADH149[3, 4] Sin[phi[i]] -
>    ADH161[3, 4] Cos[phi[2]] Sin[phi[i]] +
>    ADH161[2, 4] (Cos[phi[i]] Cos[phi[3]] - ADH149[3, 2] Sin[phi[i]]) +
>    ADH161[i, 4] (-(ADH149[3, i] Sin[phi[i]]) + Cos[phi[i]] Sin[phi[3]])
ADH160[3, 4] ->
>   ADH149[3, 4] Cos[phi[i]] + ADH161[3, 4] Cos[phi[i]] Cos[phi[2]] +
>    ADH145[2, 4] Sin[phi[i]] +
>    ADH161[2, 4] (ADH149[3, 2] Cos[phi[i]] + Cos[phi[3]] Sin[phi[i]]) +
>    ADH161[i, 4] (ADH149[3, i] Cos[phi[i]] + Sin[phi[i]] Sin[phi[3]])
ADH147[2, 3] -> -(Cos[phi[2]] Sin[phi[i]])
ADH147[3, 3] -> Cos[phi[i]] Cos[phi[2]]
ADH148[2, i] -> -(ADH149[3, i] Sin[phi[i]]) + Cos[phi[i]] Sin[phi[3]]
ADH148[3, i] -> ADH149[3, i] Cos[phi[i]] + Sin[phi[i]] Sin[phi[3]]
ADH150[2, i] -> -(ADH149[3, i] Sin[phi[i]]) + Cos[phi[i]] Sin[phi[3]]
ADH150[3, i] -> ADH149[3, i] Cos[phi[i]] + Sin[phi[i]] Sin[phi[3]]
ADH153[i, 3] -> ADH149[i, 2] ADH154[2, 3] + ADH154[3, 3] Sin[phi[2]]
ADH153[2, 3] ->
>   -(ADH154[3, 3] Cos[phi[2]] Sin[phi[i]]) +
>    ADH154[2, 3] (Cos[phi[i]] Cos[phi[3]] - ADH149[3, 2] Sin[phi[i]])
ADH153[3, 3] ->
>   ADH154[3, 3] Cos[phi[i]] Cos[phi[2]] +
>    ADH154[2, 3] (ADH149[3, 2] Cos[phi[i]] + Cos[phi[3]] Sin[phi[i]])
ADH155[i, 3] -> ADH149[i, 2] ADH156[2, 3] + ADH156[3, 3] Sin[phi[2]]
ADH155[2, 3] ->
>   -(ADH156[3, 3] Cos[phi[2]] Sin[phi[i]]) +
>    ADH156[2, 3] (Cos[phi[i]] Cos[phi[3]] - ADH149[3, 2] Sin[phi[i]])
ADH155[3, 3] ->
>   ADH156[3, 3] Cos[phi[i]] Cos[phi[2]] +
>    ADH156[2, 3] (ADH149[3, 2] Cos[phi[i]] + Cos[phi[3]] Sin[phi[i]])
ADH160[2, i] -> -(ADH149[3, i] Sin[phi[i]]) + Cos[phi[i]] Sin[phi[3]]
ADH160[3, i] -> ADH149[3, i] Cos[phi[i]] + Sin[phi[i]] Sin[phi[3]]
JACR81[4] -> ADH148[2, 4] ADH148[3, i] - ADH148[2, i] ADH148[3, 4]
JACR81[5] -> ADH148[3, 4] ADH149[i, i] - ADH148[3, i] ADH149[i, 4]
JACR81[6] -> -(ADH148[2, 4] ADH149[i, i]) + ADH148[2, i] ADH149[i, 4]
JACR82[4] -> ADH150[2, 4] ADH150[3, i] - ADH150[2, i] ADH150[3, 4]
JACR82[5] -> -(ADH150[i, 4] ADH150[3, i]) + ADH149[i, i] ADH150[3, 4]
JACR82[6] -> ADH150[i, 4] ADH150[2, i] - ADH149[i, i] ADH150[2, 4]
JACR85[4] -> ADH160[2, 4] ADH160[3, i] - ADH160[2, i] ADH160[3, 4]
JACR85[5] -> -(ADH160[i, 4] ADH160[3, i]) + ADH149[i, i] ADH160[3, 4]
JACR85[6] -> ADH160[i, 4] ADH160[2, i] - ADH149[i, i] ADH160[2, 4]
(beta[i])' -> (ADH155[3, 3] JACR81[5] (delta[i])' -
>      ADH155[2, 3] JACR81[6] (delta[i])' +
>      ADH155[3, 3] JACR82[5] (delta[3])' -
>      ADH155[2, 3] JACR82[6] (delta[3])' -
>      ADH153[3, 3] ADH155[2, 3] (s[4])' +
>      ADH153[2, 3] ADH155[3, 3] (s[4])') /
>    (ADH155[3, 3] JACR85[5] - ADH155[2, 3] JACR85[6])
(s[i])' -> (-(JACR85[5] (beta[i])') + JACR81[5] (delta[i])' +
>      JACR82[5] (delta[3])' + ADH153[2, 3] (s[4])') / ADH155[2, 3]
(beta[2])' -> (-(JACR85[4] (beta[i])') + JACR81[4] (delta[i])' +
>      JACR82[4] (delta[3])' - ADH155[i, 3] (s[i])' + ADH153[i, 3] (s[4])'
>      ) / ai
(phi[2])' -> (-(ADH147[3, 3] ADH160[2, i] (beta[i])') +
>      ADH147[2, 3] ADH160[3, i] (beta[i])' - ADH147[3, 3] (beta[2])' +
>      ADH147[3, 3] ADH148[2, i] (delta[i])' -
>      ADH147[2, 3] ADH148[3, i] (delta[i])' +
>      ADH147[3, 3] ADH150[2, i] (delta[3])' -
>      ADH147[2, 3] ADH150[3, i] (delta[3])') /
>    (ADH147[3, 3] Cos[phi[i]] - ADH147[2, 3] Sin[phi[i]])
(phi[3])' -> (-(ADH160[2, i] (beta[i])') - (beta[2])' +
>      ADH148[2, i] (delta[i])' + ADH150[2, i] (delta[3])' -
>      Cos[phi[i]] (phi[2])') / ADH147[2, 3]
(phi[i])' -> -(ADH149[i, i] (beta[i])') + ADH149[i, i] (delta[i])' +
>    ADH149[i, i] (delta[3])' - Sin[phi[2]] (phi[3])'
JACR86[6] -> ai (beta[2])'
JACR88[5] -> -(JACR85[6] (beta[i])') + JACR81[6] (delta[i])' +
>    JACR82[6] (delta[3])' - ADH155[3, 3] (s[i])' + ADH153[3, 3] (s[4])'
JACR88[6] -> JACR85[5] (beta[i])' - JACR81[5] (delta[i])' -
>    JACR82[5] (delta[3])' + ADH155[2, 3] (s[i])' - ADH153[2, 3] (s[4])'
JACR89[2] -> -(Sin[phi[i]] (phi[i])')
JACR89[3] -> Cos[phi[i]] (phi[i])'
JACR89[4] -> (Cos[phi[i]] JACR85[6] - JACR85[5] Sin[phi[i]]) (beta[i])' -
>    (Cos[phi[i]] JACR81[6] - JACR81[5] Sin[phi[i]]) (delta[i])' -
>    (Cos[phi[i]] JACR82[6] - JACR82[5] Sin[phi[i]]) (delta[3])' +
>    (ADH155[3, 3] Cos[phi[i]] - ADH155[2, 3] Sin[phi[i]]) (s[i])' -
>    (ADH153[3, 3] Cos[phi[i]] - ADH153[2, 3] Sin[phi[i]]) (s[4])'
JACR89[5] -> JACR85[4] Sin[phi[i]] (beta[i])' -
>    JACR81[4] Sin[phi[i]] (delta[i])' -
>    JACR82[4] Sin[phi[i]] (delta[3])' +
>    ADH155[i, 3] Sin[phi[i]] (s[i])' - ADH153[i, 3] Sin[phi[i]] (s[4])'
JACR89[6] -> -(Cos[phi[i]] JACR85[4] (beta[i])') +
>    Cos[phi[i]] JACR81[4] (delta[i])' +
>    Cos[phi[i]] JACR82[4] (delta[3])' -
>    ADH155[i, 3] Cos[phi[i]] (s[i])' + ADH153[i, 3] Cos[phi[i]] (s[4])'
JACR90[i] -> (ADH147[3, 3] Cos[phi[i]] - ADH147[2, 3] Sin[phi[i]])
>    (phi[2])'
JACR90[2] -> -(ADH147[3, 3] (phi[i])') + Sin[phi[i]] Sin[phi[2]] (phi[2])'
JACR90[3] -> ADH147[2, 3] (phi[i])' - Cos[phi[i]] Sin[phi[2]] (phi[2])'
JACR90[4] -> (-(ADH147[3, 3] JACR85[5]) + ADH147[2, 3] JACR85[6])
>     (beta[i])' - (-(ADH147[3, 3] JACR81[5]) + ADH147[2, 3] JACR81[6])
>     (delta[i])' - (-(ADH147[3, 3] JACR82[5]) + ADH147[2, 3] JACR82[6])
>     (delta[3])' + (-(ADH147[3, 3] ADH155[2, 3]) +
>       ADH147[2, 3] ADH155[3, 3]) (s[i])' -
>    (-(ADH147[3, 3] ADH153[2, 3]) + ADH147[2, 3] ADH153[3, 3]) (s[4])'
JACR90[5] -> (ADH147[3, 3] JACR85[4] - JACR85[6] Sin[phi[2]])
>     (beta[i])' - (ADH147[3, 3] JACR81[4] - JACR81[6] Sin[phi[2]])
```

```
>     (delta[i])' - (ADH147[3, 3] JACR82[4] - JACR82[6] Sin[phi[2]])
>     (delta[3])' + (ADH147[3, 3] ADH155[1, 3] -
>       ADH155[3, 3] Sin[phi[2]]) (s[i])' -
>    (ADH147[3, 3] ADH153[1, 3] - ADH153[3, 3] Sin[phi[2]]) (s[4])'
JACR90[6] -> (-(ADH147[2, 3] JACR85[4]) + JACR85[5] Sin[phi[2]])
>     (beta[i])' - (-(ADH147[2, 3] JACR81[4]) + JACR81[5] Sin[phi[2]])
>     (delta[i])' - (-(ADH147[2, 3] JACR82[4]) + JACR82[5] Sin[phi[2]])
>     (delta[3])' + (-(ADH147[2, 3] ADH155[1, 3]) +
>       ADH155[2, 3] Sin[phi[2]]) (s[1])' -
>    (-(ADH147[2, 3] ADH153[1, 3]) + ADH153[2, 3] Sin[phi[2]]) (s[4])'
JACR91[i] -> (ADH148[3, i] Cos[phi[i]] - ADH148[2, i] Sin[phi[i]])
>     (phi[2])' + (-(ADH147[3, 3] ADH148[2, i]) +
>       ADH147[2, 3] ADH148[3, i]) (phi[3])'
JACR91[2] -> -(ADH148[3, i] (phi[i])') +
>    ADH149[i, i] Sin[phi[i]] (phi[2])' +
>    (ADH147[3, 3] ADH149[i, i] - ADH148[3, i] Sin[phi[2]]) (phi[3])'
JACR91[3] -> ADH148[2, i] (phi[i])' -
>    ADH149[i, i] Cos[phi[i]] (phi[2])' +
>    (-(ADH147[2, 3] ADH149[i, i]) + ADH148[2, i] Sin[phi[2]]) (phi[3])'
JACR91[4] -> (-(ADH148[3, i] JACR85[5]) + ADH148[2, i] JACR81[6])
>     (beta[i])' - (-(ADH148[3, i] JACR81[5]) + ADH148[2, i] JACR81[6])
>     (delta[i])' - (-(ADH148[3, i] JACR82[5]) + ADH148[2, i] JACR82[6])
>     (delta[3])' + (Cos[phi[i]] JACR81[6] - JACR81[5] Sin[phi[i]])
>     (phi[2])' + (-(ADH147[3, 3] JACR81[5]) + ADH147[2, 3] JACR81[6])
>     (phi[3])' + (-(ADH148[3, i] ADH155[2, 3]) +
>       ADH148[2, i] ADH155[3, 3]) (s[i])' -
>    (-(ADH148[3, i] ADH153[2, 3]) + ADH148[2, i] ADH153[3, 3]) (s[4])'
JACR91[5] -> (ADH148[3, i] JACR85[6] - ADH149[i, i] JACR85[6])
>     (beta[i])' - (ADH148[3, i] JACR81[4] - ADH149[i, i] JACR81[6])
>     (delta[i])' - (ADH148[3, i] JACR82[4] - ADH149[i, i] JACR82[6])
>     (delta[3])' - JACR81[6] (phi[i])' +
>    JACR81[4] Sin[phi[i]] (phi[2])' +
>    (ADH147[3, 3] JACR81[4] - JACR81[6] Sin[phi[2]]) (phi[3])' +
>    (ADH148[3, i] ADH155[1, 3] - ADH149[i, i] ADH155[3, 3]) (s[i])' -
>    (ADH148[3, i] ADH153[1, 3] - ADH149[i, i] ADH153[3, 3]) (s[4])'
JACR91[6] -> (-(ADH148[2, i] JACR85[4]) + ADH149[i, i] JACR85[5])
>     (beta[i])' - (-(ADH148[2, i] JACR81[4]) + ADH149[i, i] JACR81[5])
>     (delta[i])' - (-(ADH148[2, i] JACR82[4]) + ADH149[i, i] JACR82[5])
>     (delta[3])' + JACR81[5] (phi[i])' -
>    Cos[phi[i]] JACR81[4] (phi[2])' +
>    (-(ADH147[2, 3] JACR81[4]) + JACR81[5] Sin[phi[2]]) (phi[3])' +
>    (-(ADH148[2, i] ADH155[i, 3]) + ADH149[i, i] ADH155[2, 3]) (s[i])' -
>    (-(ADH148[2, i] ADH153[i, 3]) + ADH149[i, i] ADH153[2, 3]) (s[4])'
JACR92[i] -> -((-(ADH148[3, i] ADH150[2, i]) + ADH148[2, i] ADH150[3, i])
>       (delta[i])') + (ADH150[3, i] Cos[phi[i]] -
>       ADH150[2, i] Sin[phi[i]]) (phi[2])' +
>    (-(ADH147[3, 3] ADH150[2, i]) + ADH147[2, 3] ADH150[3, i]) (phi[3])'
JACR92[2] -> -((ADH148[3, i] ADH149[i, i] - ADH149[i, i] ADH150[3, i])
>       (delta[i])') - ADH150[3, i] (phi[i])' +
>    ADH149[i, i] Sin[phi[i]] (phi[2])' +
>    (ADH147[3, 3] ADH149[i, i] - ADH150[3, i] Sin[phi[2]]) (phi[3])'
JACR92[3] -> -((-(ADH148[2, i] ADH149[i, i]) + ADH149[i, i] ADH150[2, i])
>       (delta[i])') + ADH150[2, i] (phi[i])' -
>    ADH149[i, i] Cos[phi[i]] (phi[2])' +
>    (-(ADH147[2, 3] ADH149[i, i]) + ADH150[2, i] Sin[phi[2]]) (phi[3])'
JACR92[4] -> (-(ADH150[3, i] JACR85[5]) + ADH150[2, i] JACR85[6])
>     (beta[i])' - (-(ADH148[3, i] JACR82[5]) + ADH148[2, i] JACR82[6])
>     (delta[i])' - (-(ADH150[3, i] JACR82[5]) + ADH150[2, i] JACR82[6])
>     (delta[3])' + (Cos[phi[i]] JACR82[6] - JACR82[5] Sin[phi[i]])
>     (phi[2])' + (-(ADH147[3, 3] JACR82[5]) + ADH147[2, 3] JACR82[6])
>     (phi[3])' + (-(ADH150[3, i] ADH155[2, 3]) +
>       ADH150[2, i] ADH155[3, 3]) (s[i])' -
>    (-(ADH150[3, i] ADH153[2, 3]) + ADH150[2, i] ADH153[3, 3]) (s[4])'
JACR92[5] -> (ADH150[3, i] JACR85[4] - ADH149[i, i] JACR85[6])
>     (beta[i])' - (ADH148[3, i] JACR82[4] - ADH149[i, i] JACR82[6])
>     (delta[i])' - (ADH150[3, i] JACR82[4] - ADH149[i, i] JACR82[6])
>     (delta[3])' - JACR82[6] (phi[i])' +
>    JACR82[4] Sin[phi[i]] (phi[2])' +
>    (ADH147[3, 3] JACR82[4] - JACR82[6] Sin[phi[2]]) (phi[3])' +
>    (ADH150[3, i] ADH155[i, 3] - ADH149[i, i] ADH155[3, 3]) (s[i])' -
>    (ADH150[3, i] ADH153[i, 3] - ADH149[i, i] ADH153[3, 3]) (s[4])'
JACR92[6] -> (-(ADH150[2, i] JACR85[4]) + ADH149[i, i] JACR85[5])
>     (beta[i])' - (-(ADH148[2, i] JACR82[4]) + ADH149[i, i] JACR82[5])
>     (delta[i])' - (-(ADH150[2, i] JACR82[4]) + ADH149[i, i] JACR82[5])
>     (delta[3])' + JACR82[5] (phi[i])' -
>    Cos[phi[i]] JACR82[4] (phi[2])' +
>    (-(ADH147[2, 3] JACR82[4]) + JACR82[5] Sin[phi[2]]) (phi[3])' +
>    (-(ADH150[2, i] ADH155[i, 3]) + ADH149[i, i] ADH155[2, 3]) (s[i])' -
>    (-(ADH150[2, i] ADH153[i, 3]) + ADH149[i, i] ADH153[2, 3]) (s[4])'
JACR93[4] -> -((-(ADH148[3, i] ADH153[2, 3]) + ADH148[2, i] ADH153[3, 3])
>       (delta[i])') - (-(ADH150[3, i] ADH153[2, 3]) +
>       ADH150[2, i] ADH153[3, 3]) (delta[3])' +
>    (ADH153[3, 3] Cos[phi[i]] - ADH153[2, 3] Sin[phi[i]]) (phi[2])' +
>    (-(ADH147[3, 3] ADH153[2, 3]) + ADH147[2, 3] ADH153[3, 3]) (phi[3])'
JACR93[5] -> -((ADH148[3, i] ADH153[i, 3] - ADH149[i, i] ADH153[3, 3])
>       (delta[i])') - (ADH150[3, i] ADH153[i, 3] -
>       ADH149[i, i] ADH153[3, 3]) (delta[3])' - ADH153[3, 3] (phi[i])' +
>    ADH153[i, 3] Sin[phi[i]] (phi[2])' +
>    (ADH147[3, 3] ADH153[i, 3] - ADH153[3, 3] Sin[phi[2]]) (phi[3])'
JACR93[6] -> -((-(ADH148[2, i] ADH153[i, 3]) + ADH149[i, i] ADH153[2, 3])
>       (delta[i])') - (-(ADH150[2, i] ADH153[i, 3]) +
>       ADH149[i, i] ADH153[2, 3]) (delta[3])' + ADH153[2, 3] (phi[i])' -
>    ADH153[i, 3] Cos[phi[i]] (phi[2])' +
>    (-(ADH147[2, 3] ADH153[i, 3]) + ADH153[2, 3] Sin[phi[2]]) (phi[3])'
JACR94[4] -> -((-(ADH148[3, i] ADH155[2, 3]) + ADH148[2, i] ADH155[3, 3])
>       (delta[i])') - (-(ADH150[3, i] ADH155[2, 3]) +
>       ADH150[2, i] ADH155[3, 3]) (delta[3])' +
>    (ADH155[3, 3] Cos[phi[i]] - ADH155[2, 3] Sin[phi[i]]) (phi[2])' +
>    (-(ADH147[3, 3] ADH155[2, 3]) + ADH147[2, 3] ADH155[3, 3]) (phi[3])'
JACR94[5] -> -((ADH148[3, i] ADH155[i, 3] - ADH149[i, i] ADH155[3, 3])
>       (delta[i])') - (ADH150[3, i] ADH155[i, 3] -
>       ADH149[i, i] ADH155[3, 3]) (delta[3])' - ADH155[3, 3] (phi[i])' +
>    ADH155[i, 3] Sin[phi[i]] (phi[2])' +
>    (ADH147[3, 3] ADH155[i, 3] - ADH155[3, 3] Sin[phi[2]]) (phi[3])'
JACR94[6] -> -((-(ADH148[2, i] ADH155[i, 3]) + ADH149[i, i] ADH155[2, 3])
>       (delta[i])') - (-(ADH150[2, i] ADH155[i, 3]) +
>       ADH149[i, i] ADH155[2, 3]) (delta[3])' + ADH155[2, 3] (phi[i])' -
>    ADH155[i, 3] Cos[phi[i]] (phi[2])' +
>    (-(ADH147[2, 3] ADH155[i, 3]) + ADH155[2, 3] Sin[phi[2]]) (phi[3])'
JACR95[i] -> -((-(ADH148[3, i] ADH160[2, i]) + ADH148[2, i] ADH160[3, i])
>       (delta[i])') - (-(ADH150[3, i] ADH160[2, i]) +
>       ADH150[2, i] ADH160[3, i]) (delta[3])' +
>    (ADH160[3, i] Cos[phi[i]] - ADH160[2, i] Sin[phi[i]]) (phi[2])' +
>    (-(ADH147[3, 3] ADH160[2, i]) + ADH147[2, 3] ADH160[3, i]) (phi[3])'
JACR95[2] -> -((ADH148[3, i] ADH149[i, i] - ADH149[i, i] ADH160[3, i])
>       (delta[i])') - (ADH149[i, i] ADH150[3, i] -
>       ADH149[i, i] ADH160[3, i]) (delta[3])' - ADH160[3, i] (phi[i])' +
>    ADH149[i, i] Sin[phi[i]] (phi[2])' +
>    (ADH147[3, 3] ADH149[i, i] - ADH160[3, i] Sin[phi[2]]) (phi[3])'
JACR95[3] -> -((-(ADH148[2, i] ADH149[i, i]) + ADH149[i, i] ADH160[2, i])
>       (delta[i])') - (-(ADH149[i, i] ADH150[2, i]) +
>       ADH149[i, i] ADH160[2, i]) (delta[3])' + ADH160[2, i] (phi[i])' -
>    ADH149[i, i] Cos[phi[i]] (phi[2])' +
>    (-(ADH147[2, 3] ADH149[i, i]) + ADH160[2, i] Sin[phi[2]]) (phi[3])'
JACR95[4] -> (-(ADH160[3, i] JACR85[5]) + ADH160[2, i] JACR85[6])
>     (beta[i])' - (-(ADH148[3, i] JACR85[5]) + ADH148[2, i] JACR85[6])
>     (delta[i])' - (-(ADH150[3, i] JACR85[5]) + ADH150[2, i] JACR85[6])
>     (delta[3])' + (Cos[phi[i]] JACR85[6] - JACR85[5] Sin[phi[i]])
>     (phi[2])' + (-(ADH147[3, 3] JACR85[5]) + ADH147[2, 3] JACR85[6])
>     (phi[3])'
JACR95[5] -> (ADH160[3, i] JACR85[4] - ADH149[i, i] JACR85[6])
>     (beta[i])' - (ADH148[3, i] JACR85[4] - ADH149[i, i] JACR85[6])
>     (delta[i])' - (ADH150[3, i] JACR85[4] - ADH149[i, i] JACR85[6])
>     (delta[3])' - JACR85[6] (phi[i])' +
>    JACR85[4] Sin[phi[i]] (phi[2])' +
>    (ADH147[3, 3] JACR85[4] - JACR85[6] Sin[phi[2]]) (phi[3])'
JACR95[6] -> (-(ADH160[2, i] JACR85[4]) + ADH149[i, i] JACR85[5])
>     (beta[i])' - (-(ADH148[2, i] JACR85[4]) + ADH149[i, i] JACR85[5])
>     (delta[i])' - (-(ADH150[2, i] JACR85[4]) + ADH149[i, i] JACR85[5])
>     (delta[3])' + JACR85[5] (phi[i])' -
>    Cos[phi[i]] JACR85[4] (phi[2])' +
>    (-(ADH147[2, 3] JACR85[4]) + JACR85[5] Sin[phi[2]]) (phi[3])'
JACR87[6] -> -(JACR86[6] (beta[2])'')
JACR96[i] -> JACR95[i] (beta[i])' - JACR91[i] (delta[i])' -
>    JACR92[i] (delta[3])' + JACR90[i] (phi[3])'
JACR96[2] -> JACR95[2] (beta[i])' - JACR91[2] (delta[i])' -
>    JACR92[2] (delta[3])' + JACR89[2] (phi[2])' + JACR90[2] (phi[3])'
JACR96[3] -> JACR95[3] (beta[i])' - JACR91[3] (delta[i])' -
>    JACR92[3] (delta[3])' + JACR89[3] (phi[2])' + JACR90[3] (phi[3])'
JACR96[4] -> JACR95[4] (beta[i])' - JACR91[4] (delta[i])' -
>    JACR92[4] (delta[3])' + JACR89[4] (phi[2])' + JACR90[4] (phi[3])' +
>    JACR94[4] (s[i])' - JACR93[4] (s[4])'
JACR96[5] -> JACR95[5] (beta[i])' - JACR91[5] (delta[i])' -
>    JACR92[5] (delta[3])' + JACR88[5] (phi[i])' + JACR89[5] (phi[2])' +
>    JACR90[5] (phi[3])' + JACR94[5] (s[i])' - JACR93[5] (s[4])'
JACR96[6] -> JACR95[6] (beta[i])' - JACR91[6] (delta[i])' -
>    JACR92[6] (delta[3])' + JACR88[6] (phi[i])' + JACR89[6] (phi[2])' +
>    JACR90[6] (phi[3])' + JACR94[6] (s[i])' - JACR93[6] (s[4])'
(beta[i])'' ->
>   (-(ADH155[2, 3] JACR87[6]) - ADH155[3, 3] JACR96[5] +
>      ADH155[2, 3] JACR96[6] + ADH155[3, 3] JACR81[5] (delta[i])'' -
>      ADH155[2, 3] JACR81[6] (delta[i])'' +
>      ADH155[3, 3] JACR82[5] (delta[3])'' -
>      ADH155[2, 3] JACR82[6] (delta[3])'' -
>      ADH153[3, 3] ADH155[2, 3] (s[4])'' +
>      ADH153[2, 3] ADH155[3, 3] (s[4])'') /
>    (ADH155[3, 3] JACR85[5] - ADH155[2, 3] JACR85[6])
(s[i])'' -> (-JACR96[5] - JACR85[5] (beta[i])'' +
>      JACR81[5] (delta[i])'' + JACR82[5] (delta[3])'' +
>      ADH153[2, 3] (s[4])'') / ADH155[2, 3]
(beta[2])'' ->
>   (-JACR96[4] - JACR85[4] (beta[i])'' + JACR81[4] (delta[i])'' +
```

```
>      JACR82[4] (delta[3])'' - ADH155[1, 3] (s[1])'' +
>      ADH153[1, 3] (s[4])'')) / a1
(phi[2])'' -> (-(ADH147[3, 3] JACR96[2]) + ADH147[2, 3] JACR96[3] -
>      ADH147[3, 3] ADH160[2, 1] (beta[1])'' +
>      ADH147[2, 3] ADH160[3, 1] (beta[1])'' - ADH147[3, 3] (beta[2])'' +
>      ADH147[3, 3] ADH148[2, 1] (delta[1])'' -
>      ADH147[2, 3] ADH148[3, 1] (delta[1])'' +
>      ADH147[3, 3] ADH150[2, 1] (delta[3])'' -
>      ADH147[2, 3] ADH150[3, 1] (delta[3])'') /
>    (ADH147[3, 3] Cos[phi[1]] - ADH147[2, 3] Sin[phi[1]])
(phi[3])'' -> (-JACR96[2] - ADH160[2, 1] (beta[1])'' - (beta[2])'' +
>      ADH148[2, 1] (delta[1])'' + ADH150[2, 1] (delta[3])'' -
>      Cos[phi[1]] (phi[2])'') / ADH147[2, 3]
(phi[1])'' -> -JACR96[1] - ADH149[1, 1] (beta[1])'' +
>    ADH149[1, 1] (delta[1])'' + ADH149[1, 1] (delta[3])'' -
>    Sin[phi[2]] (phi[3])''
beta[4] -> 0
ADH166[2, 4] -> (h1 + h2) Cos[delta[3]]
ADH163[3, 4] -> a1
ADH164[2, 4] -> -12 - 14 + ADH166[2, 4]
                  ADH164[2, 4]
beta[3] -> -ArcSin[------------]
                  ADH163[3, 4]
ADH166[3, 4] -> (h1 + h2) Sin[delta[3]]
ADH165[3, 4] -> -h3 + ADH166[3, 4] + s[4]
ADH167[2, 4] -> -(ADH163[3, 4] Sin[beta[3]])
ADH167[3, 4] -> ADH163[3, 4] Cos[beta[3]]
                                                  2
s[2] -> -(-2 ADH165[3, 4] + Sqrt[4 ADH165[3, 4]  -
                           2                2              2
>        4 (ADH164[2, 4]  + ADH165[3, 4]  - ADH167[2, 4]  -
                           2
>           ADH167[3, 4] )]) / 2
phi[6] -> -3.14159
ADH172[1, 1] -> Cos[phi[6]]
phi[5] -> -ArcTan[ADH172[1, 1], 0]
ADH175[3, 2] -> -Sin[beta[3]]
ADH175[3, 3] -> Cos[beta[3]]
ADH174[2, 2] ->
>   Cos[phi[6]] (Cos[beta[3]] Cos[delta[3]] + Sin[beta[3]] Sin[delta[3]])
ADH174[3, 2] ->
>   Cos[phi[6]] (ADH175[3, 2] Cos[delta[3]] + ADH175[3, 3] Sin[delta[3]])
phi[4] -> ArcTan[ADH174[2, 2], ADH174[3, 2]]
ADH184[2, 4] -> 12 + 14
ADH184[3, 4] -> h3 - s[4]
ADH173[1, 4] -> -((-h1 - h2) Sin[phi[6]])
ADH173[2, 4] -> (-h1 - h2) Cos[phi[6]]
ADH186[2, 4] ->
>   ADH184[2, 4] Cos[delta[3]] + ADH184[3, 4] Sin[delta[3]] +
>    s[2] Sin[delta[3]]
ADH186[3, 4] ->
>   ADH184[3, 4] Cos[delta[3]] + Cos[delta[3]] s[2] -
>    ADH184[2, 4] Sin[delta[3]]
ADH180[3, 4] -> -(ADH173[1, 4] Sin[phi[5]])
ADH179[2, 4] -> ADH173[2, 4] Cos[phi[4]] - ADH180[3, 4] Sin[phi[4]]
ADH179[3, 4] -> ADH180[3, 4] Cos[phi[4]] + ADH173[2, 4] Sin[phi[4]]
ADH180[1, 2] -> -(Cos[phi[5]] Sin[phi[6]])
ADH180[1, 4] -> ADH173[1, 4] Cos[phi[5]]
ADH180[3, 1] -> -(Cos[phi[6]] Sin[phi[5]])
ADH180[3, 2] -> Sin[phi[5]] Sin[phi[6]]
ADH185[1, 4] ->
>   ADH180[1, 4] + ADH180[1, 2] ADH186[2, 4] + ADH186[3, 4] Sin[phi[5]]
ADH185[2, 4] ->
>   ADH173[2, 4] Cos[phi[4]] - ADH180[3, 4] Sin[phi[4]] -
>    ADH186[3, 4] Cos[phi[5]] Sin[phi[4]] +
>    ADH186[2, 4] (Cos[phi[4]] Cos[phi[6]] - ADH180[3, 2] Sin[phi[4]])
ADH185[3, 4] ->
>   ADH180[3, 4] Cos[phi[4]] + ADH186[3, 4] Cos[phi[4]] Cos[phi[5]] +
>    ADH173[2, 4] Sin[phi[4]] +
>    ADH186[2, 4] (ADH180[3, 2] Cos[phi[4]] + Cos[phi[6]] Sin[phi[4]])
ADH178[2, 3] -> -(Cos[phi[5]] Sin[phi[4]])
ADH178[3, 3] -> Cos[phi[4]] Cos[phi[5]]
ADH179[2, 1] -> -(ADH180[3, 1] Sin[phi[4]]) + Cos[phi[4]] Sin[phi[6]]
ADH179[3, 1] -> ADH180[3, 1] Cos[phi[4]] + Sin[phi[4]] Sin[phi[6]]
ADH180[1, 1] -> Cos[phi[5]] Cos[phi[6]]
ADH181[1, 3] -> ADH180[3, 2] Sin[delta[3]] + Cos[delta[3]] Sin[phi[5]]
ADH181[2, 3] ->
>   -(Cos[delta[3]] Cos[phi[5]] Sin[phi[4]]) +
>    Sin[delta[3]] (Cos[phi[4]] Cos[phi[6]] - ADH180[3, 2] Sin[phi[4]])
ADH181[3, 3] ->
>   Cos[delta[3]] Cos[phi[4]] Cos[phi[5]] +
>    Sin[delta[3]] (ADH180[3, 2] Cos[phi[4]] + Cos[phi[6]] Sin[phi[4]])
ADH182[1, 3] -> ADH180[1, 2] Sin[delta[3]] + Cos[delta[3]] Sin[phi[5]]
ADH182[2, 3] ->
>   -(Cos[delta[3]] Cos[phi[5]] Sin[phi[4]]) +
>    Sin[delta[3]] (ADH180[3, 2] Cos[phi[4]] + Cos[phi[6]] Sin[phi[4]])
ADH182[3, 3] ->
>   Cos[delta[3]] Cos[phi[4]] Cos[phi[5]] +
>    Sin[delta[3]] (ADH180[3, 2] Cos[phi[4]] + Cos[phi[6]] Sin[phi[4]])
ADH185[2, 1] -> -(ADH180[3, 1] Sin[phi[4]]) + Cos[phi[4]] Sin[phi[6]]
ADH185[3, 1] -> ADH180[3, 1] Cos[phi[4]] + Sin[phi[4]] Sin[phi[6]]
JACR101[4] -> ADH179[2, 4] ADH179[3, 1] - ADH179[2, 1] ADH179[3, 4]
JACR101[5] -> ADH179[3, 4] ADH180[1, 1] - ADH179[3, 1] ADH180[1, 4]
JACR101[6] -> -(ADH179[2, 4] ADH180[1, 1]) + ADH179[2, 1] ADH180[1, 4]
JACR104[4] -> ADH185[2, 4] ADH185[3, 1] - ADH185[2, 1] ADH185[3, 4]
JACR104[5] -> -(ADH185[1, 4] ADH185[3, 1]) + ADH180[1, 1] ADH185[3, 4]
JACR104[6] -> ADH185[1, 4] ADH185[2, 1] - ADH180[1, 1] ADH185[2, 4]
(beta[3])' -> (ADH182[3, 3] JACR101[5] (delta[3])' -
>      ADH182[2, 3] JACR101[6] (delta[3])' -
>      ADH181[3, 3] ADH182[2, 3] (s[4])' +
>      ADH181[2, 3] ADH182[3, 3] (s[4])') /
>    (ADH182[3, 3] JACR104[5] - ADH182[2, 3] JACR104[6])
(s[2])' -> (-(JACR104[5] (beta[3])') + JACR101[5] (delta[3])' +
>      ADH181[2, 3] (s[4])') / ADH182[2, 3]
(beta[4])' -> (-(JACR104[4] (beta[3])') + JACR101[4] (delta[3])' -
>      ADH182[1, 3] (s[2])' + ADH181[1, 3] (s[4])')) / a1
(phi[5])' -> (-(ADH178[3, 3] ADH185[2, 1] (beta[3])') +
>      ADH178[2, 3] ADH185[3, 1] (beta[3])' +
>      ADH178[3, 3] ADH179[2, 1] (delta[3])' -
>      ADH178[2, 3] ADH179[3, 1] (delta[3])') /
>    (ADH178[3, 3] Cos[phi[4]] - ADH178[2, 3] Sin[phi[4]])
(phi[6])' -> (-(ADH185[2, 1] (beta[3])') + ADH179[2, 1] (delta[3])' -
>      Cos[phi[4]] (phi[5])') / ADH178[2, 3]
(phi[4])' -> -(ADH180[1, 1] (beta[3])') + ADH180[1, 1] (delta[3])' -
>    Sin[phi[5]] (phi[6])'
JACR107[5] -> -(JACR104[6] (beta[3])') + JACR101[6] (delta[3])' -
>    ADH182[3, 3] (s[2])' + ADH181[3, 3] (s[4])'
JACR107[6] -> JACR104[5] (beta[3])' - JACR101[5] (delta[3])' +
>    ADH182[2, 3] (s[2])' - ADH181[2, 3] (s[4])'
JACR108[2] -> -(Sin[phi[4]] (phi[4])')
JACR108[3] -> Cos[phi[4]] (phi[4])'
JACR108[4] -> (Cos[phi[4]] JACR104[6] - JACR104[5] Sin[phi[4]])
>     (beta[3])' - (Cos[phi[4]] JACR101[6] - JACR101[5] Sin[phi[4]])
>     (delta[3])' + (ADH182[3, 3] Cos[phi[4]] - ADH182[2, 3] Sin[phi[4]])
>     (s[2])' - (ADH181[3, 3] Cos[phi[4]] - ADH181[2, 3] Sin[phi[4]])
>     (s[4])'
JACR108[5] -> JACR104[4] Sin[phi[4]] (beta[3])' -
>    JACR101[4] Sin[phi[4]] (delta[3])' +
>    ADH182[1, 3] Sin[phi[4]] (s[2])' - ADH181[1, 3] Sin[phi[4]] (s[4])'
JACR108[6] -> -(Cos[phi[4]] JACR104[4] (beta[3])') +
>    Cos[phi[4]] JACR101[4] (delta[3])' -
>    ADH182[1, 3] Cos[phi[4]] (s[2])' + ADH181[1, 3] Cos[phi[4]] (s[4])'
JACR109[1] -> (ADH178[3, 3] Cos[phi[4]] - ADH178[2, 3] Sin[phi[4]])
>    (phi[5])'
JACR109[2] -> -(ADH178[3, 3] (phi[4])') +
>    Sin[phi[4]] Sin[phi[5]] (phi[5])'
JACR109[3] -> ADH178[2, 3] (phi[4])' - Cos[phi[4]] Sin[phi[5]] (phi[5])'
JACR109[4] -> (-(ADH178[3, 3] JACR104[5]) + ADH178[2, 3] JACR104[6])
>     (beta[3])' - (-(ADH178[3, 3] JACR101[5]) + ADH178[2, 3] JACR101[6])
>     (delta[3])' + (-(ADH178[3, 3] ADH182[2, 3]) +
>       ADH178[2, 3] ADH182[3, 3]) (s[2])' -
>    (-(ADH178[3, 3] ADH181[2, 3]) + ADH178[2, 3] ADH181[3, 3]) (s[4])'
JACR109[5] -> (ADH178[3, 3] JACR104[4] - JACR104[6] Sin[phi[5]])
>     (beta[3])' - (ADH178[3, 3] JACR101[4] - JACR101[6] Sin[phi[5]])
>     (delta[3])' + (ADH178[3, 3] ADH182[1, 3] -
>       ADH182[3, 3] Sin[phi[5]]) (s[2])' -
>    (ADH178[3, 3] ADH181[1, 3] - ADH181[3, 3] Sin[phi[5]]) (s[4])'
JACR109[6] -> (-(ADH178[2, 3] JACR104[4]) + JACR104[5] Sin[phi[5]])
>     (beta[3])' - (-(ADH178[2, 3] JACR101[4]) + JACR101[5] Sin[phi[5]])
>     (delta[3])' + (-(ADH178[2, 3] ADH182[1, 3]) +
>       ADH182[2, 3] Sin[phi[5]]) (s[2])' -
>    (-(ADH178[2, 3] ADH181[1, 3]) + ADH181[2, 3] Sin[phi[5]]) (s[4])'
JACR110[1] -> (ADH179[3, 1] Cos[phi[4]] - ADH179[2, 1] Sin[phi[4]])
>     (phi[5])' + (-(ADH178[3, 3] ADH179[2, 1]) +
>       ADH178[2, 3] ADH179[3, 1]) (phi[6])'
JACR110[2] -> -(ADH179[3, 1] (phi[4])') +
>    ADH180[1, 1] Sin[phi[4]] (phi[5])' +
>    (ADH178[3, 3] ADH180[1, 1] - ADH179[3, 1] Sin[phi[5]]) (phi[6])'
JACR110[3] -> ADH179[2, 1] (phi[4])' -
>    ADH180[1, 1] Cos[phi[4]] (phi[5])' +
>    (-(ADH178[2, 3] ADH180[1, 1]) + ADH179[2, 1] Sin[phi[5]]) (phi[6])'
JACR110[4] -> (-(ADH179[3, 1] JACR104[5]) + ADH179[2, 1] JACR104[6])
>     (beta[3])' - (-(ADH179[3, 1] JACR101[5]) + ADH179[2, 1] JACR101[6])
>     (delta[3])' + (Cos[phi[4]] JACR101[6] - JACR101[5] Sin[phi[4]])
>     (phi[5])' + (-(ADH178[3, 3] JACR101[5]) + ADH178[2, 3] JACR101[6])
>     (phi[6])' + (-(ADH179[3, 1] ADH182[2, 3]) +
>       ADH179[2, 1] ADH182[3, 3]) (s[2])' -
>    (-(ADH179[3, 1] ADH181[2, 3]) + ADH179[2, 1] ADH181[3, 3]) (s[4])'
JACR110[5] -> (ADH179[3, 1] JACR104[4] - ADH180[1, 1] JACR104[6])
>     (beta[3])' - (ADH179[3, 1] JACR101[6] (phi[4])' +
>    JACR101[4] Sin[phi[4]] (phi[5])' +
```

```
>    (ADH178[3, 3] JACR101[4] - JACR101[6] Sin[phi[5]]) (phi[6])' +
>    (ADH179[3, 1] ADH182[1, 3] - ADH180[1, 1] ADH182[3, 3]) (s[2])' -
>    (ADH179[3, 1] ADH181[1, 3] - ADH180[1, 1] ADH181[3, 3]) (s[4])'
JACR110[6] -> (-(ADH179[2, 1] JACR104[4]) + ADH180[1, 1] JACR104[5])
>     (beta[3])' - (-(ADH179[2, 1] JACR101[4]) + ADH180[1, 1] JACR101[5])
>     (delta[3])' + JACR101[5] (phi[4])' -
>    Cos[phi[4]] JACR101[4] (phi[5])' +
>    (-(ADH178[2, 3] JACR101[4]) + JACR101[5] Sin[phi[5]]) (phi[6])' +
>    (-(ADH179[2, 1] ADH182[1, 3]) + ADH180[1, 1] ADH182[2, 3]) (s[2])' -
>    (-(ADH179[2, 1] ADH181[1, 3]) + ADH180[1, 1] ADH181[2, 3]) (s[4])'
JACR111[4] -> -((-(ADH179[3, 1] ADH181[2, 3]) +
>         ADH179[2, 1] ADH181[3, 3]) (delta[3])') +
>    (ADH181[3, 3] Cos[phi[4]] - ADH181[2, 3] Sin[phi[4]]) (phi[5])' +
>    (-(ADH178[3, 3] ADH181[2, 3]) + ADH178[2, 3] ADH181[3, 3]) (phi[6])'
JACR111[5] -> -((ADH179[3, 1] ADH181[1, 3] - ADH180[1, 1] ADH181[3, 3])
>       (delta[3])') - ADH181[3, 3] (phi[4])' +
>    ADH181[1, 3] Sin[phi[4]] (phi[5])' +
>    (ADH178[3, 3] ADH181[1, 3] - ADH181[3, 3] Sin[phi[5]]) (phi[6])'
JACR111[6] -> -((-(ADH179[2, 1] ADH181[1, 3]) +
>         ADH180[1, 1] ADH181[2, 3]) (delta[3])') +
>    ADH181[2, 3] (phi[4])' - ADH181[1, 3] Cos[phi[4]] (phi[5])' +
>    (-(ADH178[2, 3] ADH181[1, 3]) + ADH181[2, 3] Sin[phi[5]]) (phi[6])'
JACR112[4] -> -((-(ADH179[3, 1] ADH182[2, 3]) +
>         ADH179[2, 1] ADH182[3, 3]) (delta[3])') +
>    (ADH182[3, 3] Cos[phi[4]] - ADH182[2, 3] Sin[phi[4]]) (phi[5])' +
>    (-(ADH178[3, 3] ADH182[2, 3]) + ADH178[2, 3] ADH182[3, 3]) (phi[6])'
JACR112[5] -> -((ADH179[3, 1] ADH182[1, 3] - ADH180[1, 1] ADH182[3, 3])
>       (delta[3])') - ADH182[3, 3] (phi[4])' +
>    ADH182[1, 3] Sin[phi[4]] (phi[5])' +
>    (ADH178[3, 3] ADH182[1, 3] - ADH182[3, 3] Sin[phi[5]]) (phi[6])'
JACR112[6] -> -((-(ADH179[2, 1] ADH182[1, 3]) +
>         ADH180[1, 1] ADH182[2, 3]) (delta[3])') +
>    ADH182[2, 3] (phi[4])' - ADH182[1, 3] Cos[phi[4]] (phi[5])' +
>    (-(ADH178[2, 3] ADH182[1, 3]) + ADH182[2, 3] Sin[phi[5]]) (phi[6])'
JACR113[1] -> -((-(ADH179[3, 1] ADH185[2, 1]) +
>         ADH179[2, 1] ADH185[3, 1]) (delta[3])') +
>    (ADH185[3, 1] Cos[phi[4]] - ADH185[2, 1] Sin[phi[4]]) (phi[5])' +
>    (-(ADH178[3, 3] ADH185[2, 1]) + ADH178[2, 3] ADH185[3, 1]) (phi[6])'
JACR113[2] -> -((ADH179[3, 1] ADH180[1, 1] - ADH180[1, 1] ADH185[3, 1])
>       (delta[3])') - ADH185[3, 1] (phi[4])' +
>    ADH180[1, 1] Sin[phi[4]] (phi[5])' +
>    (ADH178[3, 3] ADH180[1, 1] - ADH185[3, 1] Sin[phi[5]]) (phi[6])'
JACR113[3] -> -((-(ADH179[2, 1] ADH180[1, 1]) +
>         ADH180[1, 1] ADH185[2, 1]) (delta[3])') +
>    ADH185[2, 1] (phi[4])' - ADH180[1, 1] Cos[phi[4]] (phi[5])' +
>    (-(ADH178[2, 3] ADH180[1, 1]) + ADH185[2, 1] Sin[phi[5]]) (phi[6])'
JACR113[4] -> (-(ADH185[3, 1] JACR104[5]) + ADH185[2, 1] JACR104[6])
>     (beta[3])' - (-(ADH179[3, 1] JACR104[5]) + ADH179[2, 1] JACR104[6])
>     (delta[3])' + (Cos[phi[4]] JACR104[6] - JACR104[5] Sin[phi[4]])
>     (phi[5])' + (-(ADH178[3, 3] JACR104[5]) + ADH178[2, 3] JACR104[6])
>     (phi[6])'
JACR113[5] -> (ADH185[3, 1] JACR104[4] - ADH180[1, 1] JACR104[6])
>     (beta[3])' - (ADH179[3, 1] JACR104[4] - ADH180[1, 1] JACR104[6])
>     (delta[3])' - JACR104[6] (phi[4])' +
>    JACR104[4] Sin[phi[4]] (phi[5])' +
>    (ADH178[3, 3] JACR104[4] - JACR104[6] Sin[phi[5]]) (phi[6])'
JACR113[6] -> (-(ADH185[2, 1] JACR104[4]) + ADH180[1, 1] JACR104[5])
>     (beta[3])' - (-(ADH179[2, 1] JACR104[4]) + ADH180[1, 1] JACR104[5])
>     (delta[3])' + JACR104[5] (phi[4])' -
>    Cos[phi[4]] JACR104[4] (phi[5])' +
>    (-(ADH178[2, 3] JACR104[4]) + JACR104[5] Sin[phi[5]]) (phi[6])'
JACR114[1] -> JACR113[1] (beta[3])' - JACR110[1] (delta[3])' +
>    JACR109[1] (phi[6])'
JACR114[2] -> JACR113[2] (beta[3])' - JACR110[2] (delta[3])' +
>    JACR108[2] (phi[5])' + JACR109[2] (phi[6])'
JACR114[3] -> JACR113[3] (beta[3])' - JACR110[3] (delta[3])' +
>    JACR108[3] (phi[5])' + JACR109[3] (phi[6])'
JACR114[4] -> JACR113[4] (beta[3])' - JACR110[4] (delta[3])' +
>    JACR108[4] (phi[5])' + JACR109[4] (phi[6])' + JACR112[4] (s[2])' -
>    JACR111[4] (s[4])'
JACR114[5] -> JACR113[5] (beta[3])' - JACR110[5] (delta[3])' +
>    JACR107[5] (phi[4])' + JACR108[5] (phi[5])' + JACR109[5] (phi[6])' +
>    JACR112[5] (s[2])' - JACR111[5] (s[4])'
JACR114[6] -> JACR113[6] (beta[3])' - JACR110[6] (delta[3])' +
>    JACR107[6] (phi[4])' + JACR108[6] (phi[5])' + JACR109[6] (phi[6])' +
>    JACR112[6] (s[2])' - JACR111[6] (s[4])'
(beta[3])'' ->
>   (-(ADH182[3, 3] JACR114[5]) + ADH182[2, 3] JACR114[6] +
>      ADH182[3, 3] JACR101[5] (delta[3])'' -
>      ADH182[2, 3] JACR101[6] (delta[3])'' -
>      ADH181[3, 3] ADH182[2, 3] (s[4])'' +
>      ADH181[2, 3] ADH182[3, 3] (s[4])'') /
>    (ADH182[3, 3] JACR104[5] - ADH182[2, 3] JACR104[6])
(s[2])'' -> (-JACR114[5] - JACR104[5] (beta[3])'' +
>      JACR101[5] (delta[3])'' + ADH181[2, 3] (s[4])'') / ADH182[2, 3]
(beta[4])'' ->
>   (-JACR114[4] - JACR104[4] (beta[3])'' + JACR101[4] (delta[3])'' -
>      ADH182[1, 3] (s[2])'' + ADH181[1, 3] (s[4])'') / a1
(phi[5])'' -> (-(ADH178[3, 3] JACR114[2]) + ADH178[2, 3] JACR114[3] -
>      ADH178[3, 3] ADH185[2, 1] (beta[3])'' +
>      ADH178[2, 3] ADH185[3, 1] (beta[3])'' +
>      ADH178[3, 3] ADH179[2, 1] (delta[3])'' -
>      ADH178[2, 3] ADH179[3, 1] (delta[3])'') /
>    (ADH178[3, 3] Cos[phi[4]] - ADH178[2, 3] Sin[phi[4]])
(phi[6])'' -> (-JACR114[2] - ADH185[2, 1] (beta[3])'' +
>      ADH179[2, 1] (delta[3])'' - Cos[phi[4]] (phi[5])'') / ADH178[2, 3]
(phi[4])'' -> -JACR114[1] - ADH180[1, 1] (beta[3])'' +
>    ADH180[1, 1] (delta[3])'' - Sin[phi[5]] (phi[6])''
```

$$ADH213[3, 4] \to \frac{d\ Cos[beta[12]]}{2}$$

$$ADH217[3, 4] \to \frac{d\ Cos[beta[10]]}{2}$$

```
ADH212[2, 4] -> -(ADH213[3, 4] Sin[beta[11]])
ADH199[3, 4] -> h2 Sin[delta[3]]
                -(f Sin[delta[4]])
ADH207[3, 4] -> ------------------
                      2
ADH211[2, 4] -> -b2 + ADH212[2, 4]
ADH212[2, 1] -> Sin[beta[11]] Sin[beta[12]]
ADH212[2, 3] -> -(Cos[beta[12]] Sin[beta[11]])
ADH212[3, 1] -> -(Cos[beta[11]] Sin[beta[12]])
ADH212[3, 3] -> Cos[beta[11]] Cos[beta[12]]
ADH212[3, 4] -> ADH213[3, 4] Cos[beta[11]]
ADH216[2, 1] -> Sin[beta[9]] Sin[beta[10]]
ADH216[2, 3] -> -(Cos[beta[10]] Sin[beta[9]])
ADH216[2, 4] -> -(ADH217[3, 4] Sin[beta[9]])
ADH216[3, 1] -> -(Cos[beta[9]] Sin[beta[10]])
ADH216[3, 3] -> Cos[beta[9]] Cos[beta[10]]
ADH216[3, 4] -> ADH217[3, 4] Cos[beta[9]]
```

$$ADH191[2, 4] \to \frac{a2\ Sin[phi[10]]}{2}$$

$$ADH191[3, 4] \to \frac{a2\ Cos[phi[10]]}{2}$$

```
ADH198[3, 4] -> -h3 + ADH199[3, 4]
ADH199[2, 4] -> h2 Cos[delta[3]]
ADH206[3, 4] -> h4 + ADH207[3, 4]
                -(f Cos[delta[4]])
ADH207[2, 4] -> ------------------
                      2
ADH210[2, 1] -> ADH212[2, 1] Cos[delta[2]] - ADH212[3, 1] Sin[delta[2]]
ADH210[2, 2] -> Cos[beta[11]] Cos[delta[2]] - Sin[beta[11]] Sin[delta[2]]
ADH210[2, 3] -> ADH212[2, 3] Cos[delta[2]] - ADH212[3, 3] Sin[delta[2]]
ADH210[2, 4] -> ADH211[2, 4] Cos[delta[2]] - ADH212[3, 4] Sin[delta[2]]
ADH210[3, 1] -> ADH212[3, 1] Cos[delta[2]] + ADH212[2, 1] Sin[delta[2]]
ADH210[3, 2] -> Cos[delta[2]] Sin[beta[11]] + Cos[beta[11]] Sin[delta[2]]
ADH210[3, 3] -> ADH212[3, 3] Cos[delta[2]] + ADH212[2, 3] Sin[delta[2]]
ADH210[3, 4] -> ADH212[3, 4] Cos[delta[2]] + ADH211[2, 4] Sin[delta[2]]
```

$$ADH213[1, 4] \to \frac{d\ Sin[beta[12]]}{2}$$

```
ADH215[2, 1] -> ADH216[2, 1] Cos[delta[1]] - ADH216[3, 1] Sin[delta[1]]
ADH215[2, 2] -> Cos[beta[9]] Cos[delta[1]] - Sin[beta[9]] Sin[delta[1]]
ADH215[2, 3] -> ADH216[2, 3] Cos[delta[1]] - ADH216[3, 3] Sin[delta[1]]
ADH215[2, 4] ->
>   ADH72[2, 4] + ADH216[2, 4] Cos[delta[1]] - ADH216[3, 4] Sin[delta[1]]
ADH215[3, 1] -> ADH216[3, 1] Cos[delta[1]] + ADH216[2, 1] Sin[delta[1]]
ADH215[3, 2] -> Cos[delta[1]] Sin[beta[9]] + Cos[beta[9]] Sin[delta[1]]
ADH215[3, 3] -> ADH216[3, 3] Cos[delta[1]] + ADH216[2, 3] Sin[delta[1]]
ADH215[3, 4] ->
>   ADH72[3, 4] + ADH216[3, 4] Cos[delta[1]] + ADH216[2, 4] Sin[delta[1]]
```

$$ADH217[1, 4] \to \frac{d\ Sin[beta[10]]}{2}$$

```
ADH89[1, 3] -> -(Cos[phi[12]] Sin[phi[11]])
ADH89[2, 3] -> Sin[phi[11]] Sin[phi[12]]
ADH114[2, 3] -> -(Cos[beta[6]] Sin[beta[5]])
ADH114[3, 3] -> Cos[beta[5]] Cos[beta[6]]
ADH143[2, 3] -> -(Cos[beta[2]] Sin[beta[1]])
ADH143[3, 3] -> Cos[beta[1]] Cos[beta[2]]
ADH171[2, 3] -> -(Cos[beta[4]] Sin[beta[3]])
ADH171[3, 3] -> Cos[beta[3]] Cos[beta[4]]
ADH189[1, 2] -> -(ADH89[1, 3] Sin[phi[10]]) + Cos[phi[10]] Sin[phi[12]]
ADH189[1, 3] -> ADH89[1, 3] Cos[phi[10]] + Sin[phi[10]] Sin[phi[12]]
ADH189[1, 4] -> ADH191[3, 4] ADH89[1, 3] + ADH191[2, 4] Sin[phi[12]]
ADH189[2, 2] -> Cos[phi[10]] Cos[phi[12]] - ADH89[2, 3] Sin[phi[10]]
ADH189[2, 3] -> ADH89[2, 3] Cos[phi[10]] + Cos[phi[12]] Sin[phi[10]]
ADH189[2, 4] -> -12 + ADH191[3, 4] ADH89[2, 3] + ADH191[2, 4] Cos[phi[12]]
ADH189[3, 2] -> -(Cos[phi[11]] Sin[phi[10]])
```

```
ADH189[3, 3] -> Cos[phi[10]] Cos[phi[11]]
ADH189[3, 4] -> l3 + ADH191[3, 4] Cos[phi[11]]
ADH192[2, 4] -> -l2 - l4
                      a1 Sin[beta[2]]
ADH193[1, 4] -> l1 + ----------------
                           2
                     a1 Sin[beta[4]]
ADH194[1, 4] -> ----------------
                          2
                     a1 Sin[beta[6]]
ADH195[1, 4] -> -l1 + ----------------
                           2
ADH196[2, 4] -> -l2 - l4 + ADH199[2, 4]
ADH197[3, 4] -> ADH198[3, 4] + s[4]
ADH200[2, 2] -> Cos[delta[1]] Cos[delta[3]] - Sin[delta[1]] Sin[delta[3]]
ADH200[2, 3] ->
>   -(Cos[delta[3]] Sin[delta[1]]) - Cos[delta[1]] Sin[delta[3]]
ADH200[2, 4] -> -l2 - l4 + ADH199[2, 4]
ADH200[3, 2] -> Cos[delta[3]] Sin[delta[1]] + Cos[delta[1]] Sin[delta[3]]
ADH200[3, 3] -> Cos[delta[1]] Cos[delta[3]] - Sin[delta[1]] Sin[delta[3]]
ADH202[2, 2] -> Cos[delta[2]] Cos[delta[3]] - Sin[delta[2]] Sin[delta[3]]
ADH202[2, 3] ->
>   -(Cos[delta[3]] Sin[delta[2]]) - Cos[delta[2]] Sin[delta[3]]
ADH202[2, 4] -> -l2 - l4 + ADH199[2, 4]
ADH202[3, 2] -> Cos[delta[3]] Sin[delta[2]] + Cos[delta[2]] Sin[delta[3]]
ADH202[3, 3] -> Cos[delta[2]] Cos[delta[3]] - Sin[delta[2]] Sin[delta[3]]
ADH204[2, 4] -> -l2 - l4 + ADH207[2, 4]
ADH205[3, 4] -> ADH206[3, 4] + s[4]
ADH208[2, 1] -> ADH210[2, 1] Cos[delta[3]] - ADH210[3, 1] Sin[delta[3]]
ADH208[2, 2] -> ADH210[2, 2] Cos[delta[3]] - ADH210[3, 2] Sin[delta[3]]
ADH208[2, 3] -> ADH210[2, 3] Cos[delta[3]] - ADH210[3, 3] Sin[delta[3]]
ADH208[2, 4] ->
>   -l2 - l4 + ADH199[2, 4] + ADH210[2, 4] Cos[delta[3]] -
>    ADH210[3, 4] Sin[delta[3]]
ADH208[3, 1] -> ADH210[3, 1] Cos[delta[3]] + ADH210[2, 1] Sin[delta[3]]
ADH208[3, 2] -> ADH210[3, 2] Cos[delta[3]] + ADH210[2, 2] Sin[delta[3]]
ADH208[3, 3] -> ADH210[3, 3] Cos[delta[3]] + ADH210[2, 3] Sin[delta[3]]
ADH209[1, 4] -> -g + ADH213[1, 4]
ADH214[2, 1] -> ADH215[2, 1] Cos[delta[3]] - ADH215[3, 1] Sin[delta[3]]
ADH214[2, 2] -> ADH215[2, 2] Cos[delta[3]] - ADH215[3, 2] Sin[delta[3]]
ADH214[2, 3] -> ADH215[2, 3] Cos[delta[3]] - ADH215[3, 3] Sin[delta[3]]
ADH214[2, 4] ->
>   -l2 - l4 + ADH199[2, 4] + ADH215[2, 4] Cos[delta[3]] -
>    ADH215[3, 4] Sin[delta[3]]
ADH214[3, 1] -> ADH215[3, 1] Cos[delta[3]] + ADH215[2, 1] Sin[delta[3]]
ADH214[3, 2] -> ADH215[3, 2] Cos[delta[3]] + ADH215[2, 2] Sin[delta[3]]
ADH214[3, 3] -> ADH215[3, 3] Cos[delta[3]] + ADH215[2, 3] Sin[delta[3]]
ADH215[1, 4] -> g + ADH217[1, 4]
ADH218[2, 1] -> Sin[delta[4]] Sin[delta[5]]
ADH218[2, 3] -> -(Cos[delta[5]] Sin[delta[4]])
                                        f Cos[delta[4]]
ADH218[2, 4] -> -l2 - l4 + ADH207[2, 4] - ----------------
                                             2
ADH218[3, 1] -> -(Cos[delta[4]] Sin[delta[5]])
ADH218[3, 3] -> Cos[delta[4]] Cos[delta[5]]
ADH89[1, 1] -> Cos[phi[11]] Cos[phi[12]]
ADH89[2, 1] -> -(Cos[phi[11]] Sin[phi[12]])
                     a1 ADH143[2, 3]
ADH193[2, 4] -> ----------------
                           2
                     a1 ADH143[3, 3]
ADH193[3, 4] -> ---------------- + s[1]
                           2
                     a1 ADH171[2, 3]
ADH194[2, 4] -> ----------------
                           2
                     a1 ADH171[3, 3]
ADH194[3, 4] -> ---------------- + s[2]
                           2
                     a1 ADH114[2, 3]
ADH195[2, 4] -> ----------------
                           2
                     a1 ADH114[3, 3]
ADH195[3, 4] -> ---------------- + s[3]
                           2
ADH208[3, 4] ->
>   ADH197[3, 4] + ADH210[3, 4] Cos[delta[3]] + ADH210[2, 4] Sin[delta[3]]
ADH214[3, 4] ->
>   ADH197[3, 4] + ADH215[3, 4] Cos[delta[3]] + ADH215[2, 4] Sin[delta[3]]
                                  f Sin[delta[4]]
ADH218[3, 4] -> ADH205[3, 4] - ----------------
                                       2

R2[1, 1] -> 1
R2[1, 2] -> 0
R2[1, 3] -> 0
R2[2, 1] -> 0
R2[2, 2] -> 1
R2[2, 3] -> 0
R2[3, 1] -> 0
R2[3, 2] -> 0
R2[3, 3] -> 1
R3[1, 1] -> 1
R3[1, 2] -> 0
R3[1, 3] -> 0
R3[2, 1] -> 0
R3[2, 2] -> 1
R3[2, 3] -> 0
R3[3, 1] -> 0
R3[3, 2] -> 0
R3[3, 3] -> 1
R4[1, 1] -> 1
R4[1, 2] -> 0
R4[1, 3] -> 0
R4[2, 1] -> 0
R4[2, 2] -> 1
R4[2, 3] -> 0
R4[3, 1] -> 0
R4[3, 2] -> 0
R4[3, 3] -> 1
R5[1, 1] -> ADH89[1, 1]
R5[1, 2] -> ADH189[1, 2]
R5[1, 3] -> ADH189[1, 3]
R5[2, 1] -> ADH89[2, 1]
R5[2, 2] -> ADH189[2, 2]
R5[2, 3] -> ADH189[2, 3]
R5[3, 1] -> Sin[phi[11]]
R5[3, 2] -> ADH189[3, 2]
R5[3, 3] -> ADH189[3, 3]
R6[1, 1] -> 1
R6[1, 2] -> 0
R6[1, 3] -> 0
R6[2, 1] -> 0
R6[2, 2] -> 1
R6[2, 3] -> 0
R6[3, 1] -> 0
R6[3, 2] -> 0
R6[3, 3] -> 1
R7[1, 1] -> Cos[beta[2]]
R7[1, 2] -> 0
R7[1, 3] -> Sin[beta[2]]
R7[2, 1] -> ADH143[2, 1]
R7[2, 2] -> Cos[beta[1]]
R7[2, 3] -> ADH143[2, 3]
R7[3, 1] -> ADH143[3, 1]
R7[3, 2] -> Sin[beta[1]]
R7[3, 3] -> ADH143[3, 3]
R8[1, 1] -> Cos[beta[4]]
R8[1, 2] -> 0
R8[1, 3] -> Sin[beta[4]]
R8[2, 1] -> ADH171[2, 1]
R8[2, 2] -> Cos[beta[3]]
R8[2, 3] -> ADH171[2, 3]
R8[3, 1] -> ADH171[3, 1]
R8[3, 2] -> Sin[beta[3]]
R8[3, 3] -> ADH171[3, 3]
R9[1, 1] -> Cos[beta[6]]
R9[1, 2] -> 0
R9[1, 3] -> Sin[beta[6]]
R9[2, 1] -> ADH114[2, 1]
R9[2, 2] -> Cos[beta[5]]
R9[2, 3] -> ADH114[2, 3]
R9[3, 1] -> ADH114[3, 1]
R9[3, 2] -> Sin[beta[5]]
R9[3, 3] -> ADH114[3, 3]
R11[1, 1] -> 1
R11[1, 2] -> 0
R11[1, 3] -> 0
R11[2, 1] -> 0
R11[2, 2] -> Cos[delta[3]]
R11[2, 3] -> -Sin[delta[3]]
R11[3, 1] -> 0
R11[3, 2] -> Sin[delta[3]]
R11[3, 3] -> Cos[delta[3]]
R10[1, 1] -> 1
R10[1, 2] -> 0
R10[1, 3] -> 0
R10[2, 1] -> 0
R10[2, 2] -> ADH200[2, 2]
R10[2, 3] -> ADH200[2, 3]
R10[3, 1] -> 0
R10[3, 2] -> ADH200[3, 2]
R10[3, 3] -> ADH200[3, 3]
```

```
R12[1, 1] -> 1
R12[1, 2] -> 0
R12[1, 3] -> 0
R12[2, 1] -> 0
R12[2, 2] -> ADH202[2, 2]
R12[2, 3] -> ADH202[2, 3]
R12[3, 1] -> 0
R12[3, 2] -> ADH202[3, 2]
R12[3, 3] -> ADH202[3, 3]
R13[1, 1] -> 1
R13[1, 2] -> 0
R13[1, 3] -> 0
R13[2, 1] -> 0
R13[2, 2] -> Cos[delta[4]]
R13[2, 3] -> -Sin[delta[4]]
R13[3, 1] -> 0
R13[3, 2] -> Sin[delta[4]]
R13[3, 3] -> Cos[delta[4]]
R14[1, 1] -> Cos[beta[12]]
R14[1, 2] -> 0
R14[1, 3] -> Sin[beta[12]]
R14[2, 1] -> ADH208[2, 1]
R14[2, 2] -> ADH208[2, 2]
R14[2, 3] -> ADH208[2, 3]
R14[3, 1] -> ADH208[3, 1]
R14[3, 2] -> ADH208[3, 2]
R14[3, 3] -> ADH208[3, 3]
R15[1, 1] -> Cos[beta[10]]
R15[1, 2] -> 0
R15[1, 3] -> Sin[beta[10]]
R15[2, 1] -> ADH214[2, 1]
R15[2, 2] -> ADH214[2, 2]
R15[2, 3] -> ADH214[2, 3]
R15[3, 1] -> ADH214[3, 1]
R15[3, 2] -> ADH214[3, 2]
R15[3, 3] -> ADH214[3, 3]
R16[1, 1] -> Cos[delta[5]]
R16[1, 2] -> 0
R16[1, 3] -> Sin[delta[5]]
R16[2, 1] -> ADH218[2, 1]
R16[2, 2] -> Cos[delta[4]]
R16[2, 3] -> ADH218[2, 3]
R16[3, 1] -> ADH218[3, 1]
R16[3, 2] -> Sin[delta[4]]
R16[3, 3] -> ADH218[3, 3]
r2[1] -> 11
r2[2] -> 0
r2[3] -> s[1]
r3[1] -> 0
r3[2] -> 0
r3[3] -> s[2]
r4[1] -> -11
r4[2] -> 0
r4[3] -> s[3]
r5[1] -> ADH189[1, 4] ADH89[1, 1] + ADH189[2, 4] ADH89[2, 1] +
>    ADH189[3, 4] Sin[phi[11]]
r5[2] -> ADH189[1, 2] ADH189[1, 4] + ADH189[2, 2] ADH189[2, 4] +
>    ADH189[3, 2] ADH189[3, 4]
r5[3] -> ADH189[1, 3] ADH189[1, 4] + ADH189[2, 3] ADH189[2, 4] +
>    ADH189[3, 3] ADH189[3, 4]
r6[1] -> 0
r6[2] -> ADH192[2, 4]
r6[3] -> s[4]
r7[1] -> ADH143[2, 1] ADH193[2, 4] + ADH143[3, 1] ADH193[3, 4] +
>    ADH193[1, 4] Cos[beta[2]]
r7[2] -> ADH193[2, 4] Cos[beta[1]] + ADH193[3, 4] Sin[beta[1]]
r7[3] -> ADH143[2, 3] ADH193[2, 4] + ADH143[3, 3] ADH193[3, 4] +
>    ADH193[1, 4] Sin[beta[2]]
r8[1] -> ADH171[2, 1] ADH194[2, 4] + ADH171[3, 1] ADH194[3, 4] +
>    ADH194[1, 4] Cos[beta[4]]
r8[2] -> ADH194[2, 4] Cos[beta[3]] + ADH194[3, 4] Sin[beta[3]]
r8[3] -> ADH171[2, 3] ADH194[2, 4] + ADH171[3, 3] ADH194[3, 4] +
>    ADH194[1, 4] Sin[beta[4]]
r9[1] -> ADH114[2, 1] ADH195[2, 4] + ADH114[3, 1] ADH195[3, 4] +
>    ADH195[1, 4] Cos[beta[6]]
r9[2] -> ADH195[2, 4] Cos[beta[5]] + ADH195[3, 4] Sin[beta[5]]
r9[3] -> ADH114[2, 3] ADH195[2, 4] + ADH114[3, 3] ADH195[3, 4] +
>    ADH195[1, 4] Sin[beta[6]]
r11[1] -> 0
r11[2] -> ADH196[2, 4] Cos[delta[3]] + ADH197[3, 4] Sin[delta[3]]
r11[3] -> ADH197[3, 4] Cos[delta[3]] - ADH196[2, 4] Sin[delta[3]]
r10[1] -> g
r10[2] -> ADH200[2, 2] ADH200[2, 4] + ADH197[3, 4] ADH200[3, 2]
r10[3] -> ADH200[2, 3] ADH200[2, 4] + ADH197[3, 4] ADH200[3, 3]
r12[1] -> -g
r12[2] -> ADH202[2, 2] ADH202[2, 4] + ADH197[3, 4] ADH202[3, 2]
r12[3] -> ADH202[2, 3] ADH202[2, 4] + ADH197[3, 4] ADH202[3, 3]
r13[1] -> 0
r13[2] -> ADH204[2, 4] Cos[delta[4]] + ADH205[3, 4] Sin[delta[4]]
r13[3] -> ADH205[3, 4] Cos[delta[4]] - ADH204[2, 4] Sin[delta[4]]
r14[1] -> ADH208[2, 1] ADH208[2, 4] + ADH208[3, 1] ADH208[3, 4] +
>    ADH209[1, 4] Cos[beta[12]]
r14[2] -> ADH208[2, 2] ADH208[2, 4] + ADH208[3, 2] ADH208[3, 4]
r14[3] -> ADH208[2, 3] ADH208[2, 4] + ADH208[3, 3] ADH208[3, 4] +
>    ADH209[1, 4] Sin[beta[12]]
r15[1] -> ADH214[2, 1] ADH214[2, 4] + ADH214[3, 1] ADH214[3, 4] +
>    ADH215[1, 4] Cos[beta[10]]
r15[2] -> ADH214[2, 2] ADH214[2, 4] + ADH214[3, 2] ADH214[3, 4]
r15[3] -> ADH214[2, 3] ADH214[2, 4] + ADH214[3, 3] ADH214[3, 4] +
>    ADH215[1, 4] Sin[beta[10]]
r16[1] -> ADH218[2, 1] ADH218[2, 4] + ADH218[3, 1] ADH218[3, 4]
r16[2] -> ADH218[2, 4] Cos[delta[4]] + ADH218[3, 4] Sin[delta[4]]
r16[3] -> ADH218[2, 3] ADH218[2, 4] + ADH218[3, 3] ADH218[3, 4]
ADH18[3, 4] -> -(b2 Sin[beta[11]])
ADH49[3, 4] -> -(b2 Sin[beta[9]])
ADH17[1, 4] -> -(ADH18[3, 4] Sin[beta[12]])
ADH17[3, 4] -> ADH18[3, 4] Cos[beta[12]]
ADH18[2, 4] -> b2 Cos[beta[11]]
ADH245[3, 4] -> -(b2 Sin[beta[9]])
ADH48[1, 4] -> -(g Cos[beta[10]]) - ADH49[3, 4] Sin[beta[10]]
ADH48[3, 4] -> ADH49[3, 4] Cos[beta[10]] - g Sin[beta[10]]
ADH49[2, 4] -> b2 Cos[beta[9]]
ADH244[3, 4] -> ADH245[3, 4] Cos[beta[10]]
                -d
ADH238[3, 4] -> -- + ADH36[3, 4]
                2
ADH239[1, 4] -> -(h2 ADH17[1, 2]) + ADH17[1, 4] + g Cos[beta[12]]
ADH239[2, 4] -> -(h2 ADH18[2, 2]) + ADH18[2, 4]
                -d
ADH239[3, 4] -> -- - h2 ADH17[3, 2] + ADH17[3, 4] + g Sin[beta[12]]
                2
                -d
ADH243[3, 4] -> -- + ADH244[3, 4]
                2
ADH244[1, 4] -> -(ADH245[3, 4] Sin[beta[10]])
ADH245[2, 4] -> b2 Cos[beta[9]]
ADH246[1, 4] -> -(h2 ADH48[1, 2]) + ADH48[1, 4]
ADH246[2, 4] -> -(h2 ADH49[2, 2]) + ADH49[2, 4]
                -d
ADH246[3, 4] -> -- - h2 ADH48[3, 2] + ADH48[3, 4]
                2
ADH224[1, 3] -> -(Cos[beta[1]] Sin[beta[2]])
ADH224[3, 3] -> Cos[beta[1]] Cos[beta[2]]
ADH227[1, 3] -> -(Cos[beta[3]] Sin[beta[4]])
ADH227[3, 3] -> Cos[beta[3]] Cos[beta[4]]
ADH230[1, 3] -> -(Cos[beta[5]] Sin[beta[6]])
ADH230[3, 3] -> Cos[beta[5]] Cos[beta[6]]
ADH233[2, 3] -> Cos[delta[3]] Sin[delta[1]] + Cos[delta[1]] Sin[delta[3]]
ADH233[3, 3] -> Cos[delta[1]] Cos[delta[3]] - Sin[delta[1]] Sin[delta[3]]
ADH234[2, 3] -> Cos[delta[3]] Sin[delta[2]] + Cos[delta[2]] Sin[delta[3]]
ADH234[3, 3] -> Cos[delta[2]] Cos[delta[3]] - Sin[delta[2]] Sin[delta[3]]
ADH240[1, 3] -> ADH17[1, 3] Cos[delta[3]] + ADH17[1, 2] Sin[delta[3]]
ADH240[2, 3] -> ADH18[2, 3] Cos[delta[3]] + ADH18[2, 2] Sin[delta[3]]
ADH240[3, 3] -> ADH17[3, 3] Cos[delta[3]] + ADH17[3, 2] Sin[delta[3]]
ADH247[1, 3] -> ADH48[1, 3] Cos[delta[3]] + ADH48[1, 2] Sin[delta[3]]
ADH247[2, 3] -> ADH49[2, 3] Cos[delta[3]] + ADH49[2, 2] Sin[delta[3]]
ADH247[3, 3] -> ADH48[3, 3] Cos[delta[3]] + ADH48[3, 2] Sin[delta[3]]
ADH250[1, 3] -> -(Cos[delta[4]] Sin[delta[5]])
ADH250[3, 3] -> Cos[delta[4]] Cos[delta[5]]
             a2 Cos[phi[10]]
JACR119[4] -> ---------------
                   2
             a2 ADH90[2, 3]
JACR120[4] -> -------------
                   2
             -(a2 Sin[phi[11]])
JACR120[5] -> ------------------
                   2
             -(a1 Cos[beta[2]])
JACR123[5] -> ------------------
                   2
             -(a1 Cos[beta[4]])
JACR126[5] -> ------------------
                   2
             -(a1 Cos[beta[6]])
JACR129[5] -> ------------------
                   2
             -(d Cos[beta[12]])
JACR142[5] -> ------------------
                   2
JACR143[4] -> ADH37[2, 4] Sin[beta[12]]
JACR143[5] -> ADH238[3, 4] Cos[beta[12]] - ADH36[1, 4] Sin[beta[12]]
```

```
JACR143[6] -> -(ADH37[2, 4] Cos[beta[12]])
JACR144[4] -> ADH239[2, 4] Sin[beta[12]]
JACR144[5] -> ADH239[3, 4] Cos[beta[12]] - ADH239[1, 4] Sin[beta[12]]
JACR144[6] -> -(ADH239[2, 4] Cos[beta[12]])
              -(d Cos[beta[10]])
JACR147[5] -> ------------------
                      2
JACR148[4] -> ADH245[2, 4] Sin[beta[10]]
JACR148[5] -> ADH243[3, 4] Cos[beta[10]] - ADH244[1, 4] Sin[beta[10]]
JACR148[6] -> -(ADH245[2, 4] Cos[beta[10]])
JACR149[4] -> ADH246[2, 4] Sin[beta[10]]
JACR149[5] -> ADH246[3, 4] Cos[beta[10]] - ADH246[1, 4] Sin[beta[10]]
JACR149[6] -> -(ADH246[2, 4] Cos[beta[10]])
JACR152[4] -> f Sin[delta[5]]
JACR152[6] -> -(f Cos[delta[5]])
omega2[1] -> 0
omega2[2] -> 0
omega2[3] -> 0
omega3[1] -> 0
omega3[2] -> 0
omega3[3] -> 0
omega4[1] -> 0
omega4[2] -> 0
omega4[3] -> 0
omega5[1] -> -(phi[10])' - Sin[phi[11]] (phi[12])'
omega5[2] -> -(Cos[phi[10]] (phi[11])') - ADH90[2, 3] (phi[12])'
omega5[3] -> -(Sin[phi[10]] (phi[11])') - ADH90[3, 3] (phi[12])'
omega6[1] -> 0
omega6[2] -> 0
omega6[3] -> 0
omega7[1] -> Cos[beta[2]] (beta[1])'
omega7[2] -> (beta[2])'
omega7[3] -> Sin[beta[2]] (beta[1])'
omega8[1] -> Cos[beta[4]] (beta[3])'
omega8[2] -> (beta[4])'
omega8[3] -> Sin[beta[4]] (beta[3])'
omega9[1] -> Cos[beta[6]] (beta[5])'
omega9[2] -> (beta[6])'
omega9[3] -> Sin[beta[6]] (beta[5])'
omega11[1] -> (delta[3])'
omega11[2] -> 0
omega11[3] -> 0
omega10[1] -> (delta[1])' + (delta[3])'
omega10[2] -> 0
omega10[3] -> 0
omega12[1] -> (delta[2])' + (delta[3])'
omega12[2] -> 0
omega12[3] -> 0
omega13[1] -> (delta[4])'
omega13[2] -> 0
omega13[3] -> 0
omega14[1] -> Cos[beta[12]] ((beta[11])' + (delta[2])' + (delta[3])')
omega14[2] -> (beta[12])'
omega14[3] -> Sin[beta[12]] ((beta[11])' + (delta[2])' + (delta[3])')
omega15[1] -> Cos[beta[10]] ((beta[9])' + (delta[1])' + (delta[3])')
omega15[2] -> (beta[10])'
omega15[3] -> Sin[beta[10]] ((beta[9])' + (delta[1])' + (delta[3])')
omega16[1] -> Cos[delta[5]] (delta[4])'
omega16[2] -> (delta[5])'
omega16[3] -> Sin[delta[5]] (delta[4])'
v2[1] -> 0
v2[2] -> 0
v2[3] -> (s[1])'
v3[1] -> 0
v3[2] -> 0
v3[3] -> (s[2])'
v4[1] -> 0
v4[2] -> 0
v4[3] -> (s[3])'
v5[1] -> -(JACR119[4] (phi[11])') - JACR120[4] (phi[12])'
         a2 (phi[10])'
v5[2] -> ------------ - JACR120[5] (phi[12])'
                2
v5[3] -> 0
v6[1] -> 0
v6[2] -> 0
v6[3] -> (s[4])'
         a1 (beta[2])'
v7[1] -> ------------ + ADH224[1, 3] (s[1])'
                2
v7[2] -> JACR123[5] (beta[1])' + Sin[beta[1]] (s[1])'
v7[3] -> ADH224[3, 3] (s[1])'
         a1 (beta[4])'
v8[1] -> ------------ + ADH227[1, 3] (s[2])'
                2
v8[2] -> JACR126[5] (beta[3])' + Sin[beta[3]] (s[2])'
v8[3] -> ADH227[3, 3] (s[2])'
         a1 (beta[6])'
v9[1] -> ------------ + ADH230[1, 3] (s[3])'
                2
v9[2] -> JACR129[5] (beta[5])' + Sin[beta[5]] (s[3])'
v9[3] -> ADH230[3, 3] (s[3])'
v11[1] -> 0
v11[2] -> Sin[delta[3]] (s[4])'
v11[3] -> h2 (delta[3])' + Cos[delta[3]] (s[4])'
v10[1] -> 0
v10[2] -> ADH151[3, 4] (delta[3])' + ADH233[2, 3] (s[4])'
v10[3] -> -(ADH151[2, 4] (delta[3])') + ADH233[3, 3] (s[4])'
v12[1] -> 0
v12[2] -> ADH109[3, 4] (delta[3])' + ADH234[2, 3] (s[4])'
v12[3] -> -(ADH109[2, 4] (delta[3])') + ADH234[3, 3] (s[4])'
v13[1] -> 0
v13[2] -> Sin[delta[4]] (s[4])'
          -(f (delta[4])')
v13[3] -> ---------------- + Cos[delta[4]] (s[4])'
                  2
          d (beta[12])'
v14[1] -> ------------ + JACR143[4] (delta[2])' +
                2
>    JACR144[4] (delta[3])' + ADH240[1, 3] (s[4])'
v14[2] -> JACR142[5] (beta[11])' + JACR143[5] (delta[2])' +
>    JACR144[5] (delta[3])' + ADH240[2, 3] (s[4])'
v14[3] -> JACR143[6] (delta[2])' + JACR144[6] (delta[3])' +
>    ADH240[3, 3] (s[4])'
         d (beta[10])'
v15[1] -> ------------ + JACR148[4] (delta[1])' +
                2
>    JACR149[4] (delta[3])' + ADH247[1, 3] (s[4])'
v15[2] -> JACR147[5] (beta[9])' + JACR148[5] (delta[1])' +
>    JACR149[5] (delta[3])' + ADH247[2, 3] (s[4])'
v15[3] -> JACR148[6] (delta[1])' + JACR149[6] (delta[3])' +
>    ADH247[3, 3] (s[4])'
v16[1] -> JACR152[4] (delta[4])' + ADH250[1, 3] (s[4])'
v16[2] -> Sin[delta[4]] (s[4])'
v16[3] -> JACR152[6] (delta[4])' + ADH250[3, 3] (s[4])'
          -(a2 ADH90[3, 3] (phi[10])')
JACR160[4] -> --------------------------- +
                        2
>    ADH90[3, 3] JACR120[5] (phi[12])'
JACR160[5] -> -(ADH90[3, 3] JACR119[4] (phi[11])') -
>    ADH90[3, 3] JACR120[4] (phi[12])'
          a2 Sin[phi[11]] (phi[10])'
JACR160[6] -> --------------------------- +
                        2
>    ADH90[2, 3] JACR119[4] (phi[11])' -
>    (-(ADH90[2, 3] JACR120[4]) + JACR120[5] Sin[phi[11]]) (phi[12])'
JACR161[1] -> -((-(ADH90[3, 3] Cos[phi[10]]) + ADH90[2, 3] Sin[phi[10]])
>      (phi[12])')
JACR161[2] -> Sin[phi[10]] Sin[phi[11]] (phi[12])'
JACR161[3] -> -(Cos[phi[10]] Sin[phi[11]] (phi[12])')
          -(a2 Sin[phi[10]] (phi[10])')
JACR161[4] -> ----------------------------
                        2
JACR161[5] -> -(JACR119[4] Sin[phi[10]] (phi[11])') -
>    ADH90[3, 3] JACR119[4] (phi[12])'
JACR161[6] -> Cos[phi[10]] JACR119[4] (phi[11])' +
>    ADH90[2, 3] JACR119[4] (phi[12])'
JACR162[2] -> -(Sin[phi[10]] (phi[11])') - ADH90[3, 3] (phi[12])'
JACR162[3] -> Cos[phi[10]] (phi[11])' + ADH90[2, 3] (phi[12])'
              -(a2 Sin[phi[10]] (phi[11])')   a2 ADH90[3, 3] (phi[12])'
JACR162[4] -> ---------------------------- - ------------------------
                        2                              2
          a2 (phi[10])'   a2 Sin[phi[11]] (phi[12])'
JACR162[6] -> ------------ + -------------------------
                  2                    2
JACR167[4] -> -(JACR123[5] Sin[beta[2]] (beta[1])')
          a1 Sin[beta[2]] (beta[2])'
JACR167[5] -> ---------------------------
                        2
JACR167[6] -> Cos[beta[2]] JACR123[5] (beta[1])'
JACR168[1] -> -(Sin[beta[2]] (beta[1])')
JACR168[3] -> Cos[beta[2]] (beta[1])'
          a1 Sin[beta[2]] (beta[1])'
JACR168[5] -> ---------------------------
                        2
          -(a1 (beta[2])')
JACR168[6] -> ----------------
                  2
JACR171[4] -> -(JACR126[5] Sin[beta[4]] (beta[3])')
          a1 Sin[beta[4]] (beta[4])'
JACR171[5] -> ---------------------------
                        2
```

```
JACR171[6] -> Cos[beta[4]] JACR126[5] (beta[3])'
JACR172[1] -> -(Sin[beta[4]] (beta[3])')
JACR172[3] -> Cos[beta[4]] (beta[3])'
              af Sin[beta[4]] (beta[3])'
JACR172[5] -> --------------------------
                          2
              -(af (beta[4])')
JACR172[6] -> ----------------
                     2
JACR175[4] -> -(JACR129[5] Sin[beta[6]] (beta[5])')
              af Sin[beta[6]] (beta[6])'
JACR175[5] -> --------------------------
                          2
JACR175[6] -> Cos[beta[6]] JACR129[5] (beta[5])'
JACR176[1] -> -(Sin[beta[6]] (beta[5])')
JACR176[3] -> Cos[beta[6]] (beta[5])'
              af Sin[beta[6]] (beta[5])'
JACR176[5] -> --------------------------
                          2
              -(af (beta[6])')
JACR176[6] -> ----------------
                     2
JACR179[5] -> -(h2 (delta[3])')
JACR182[5] -> ADH151[2, 4] (delta[3])'
JACR182[6] -> ADH151[3, 4] (delta[3])'
JACR186[5] -> ADH109[2, 4] (delta[3])'
JACR186[6] -> ADH109[3, 4] (delta[3])'
              f (delta[4])'
JACR190[5] -> -------------
                        2
JACR193[4] -> -(JACR142[5] Sin[beta[12]] (beta[11])') -
>     JACR143[5] Sin[beta[12]] (delta[2])' -
>     JACR144[5] Sin[beta[12]] (delta[3])'
              d Sin[beta[12]] (beta[12])'
JACR193[5] -> --------------------------- +
                          2
>     (-(Cos[beta[12]] JACR143[6]) + JACR143[4] Sin[beta[12]])
>      (delta[2])' + (-(Cos[beta[12]] JACR144[6]) +
>        JACR144[4] Sin[beta[12]]) (delta[3])'
JACR193[6] -> Cos[beta[12]] JACR142[5] (beta[11])' +
>     Cos[beta[12]] JACR143[5] (delta[2])' +
>     Cos[beta[12]] JACR144[5] (delta[3])'
JACR194[4] -> -(JACR142[5] Sin[beta[12]] (beta[11])') -
>     JACR143[5] Sin[beta[12]] (delta[2])' -
>     JACR143[5] Sin[beta[12]] (delta[3])'
              d Sin[beta[12]] (beta[12])'
JACR194[5] -> --------------------------- +
                          2
>     (-(Cos[beta[12]] JACR143[6]) + JACR143[4] Sin[beta[12]])
>      (delta[2])' + (-(Cos[beta[12]] JACR143[6]) +
>        JACR143[4] Sin[beta[12]]) (delta[3])'
JACR194[6] -> Cos[beta[12]] JACR142[5] (beta[11])' +
>     Cos[beta[12]] JACR143[5] (delta[2])' +
>     Cos[beta[12]] JACR143[5] (delta[3])'
JACR195[4] -> -(JACR142[5] Sin[beta[12]] (beta[11])') -
>     JACR142[5] Sin[beta[12]] (delta[2])' -
>     JACR142[5] Sin[beta[12]] (delta[3])'
              d Sin[beta[12]] (beta[12])'
JACR195[5] -> --------------------------
                          2
JACR195[6] -> Cos[beta[12]] JACR142[5] (beta[11])' +
>     Cos[beta[12]] JACR142[5] (delta[2])' +
>     Cos[beta[12]] JACR142[5] (delta[3])'
JACR196[1] -> -(Sin[beta[12]] (beta[11])') - Sin[beta[12]] (delta[2])' -
>     Sin[beta[12]] (delta[3])'
JACR196[3] -> Cos[beta[12]] (beta[11])' + Cos[beta[12]] (delta[2])' +
>     Cos[beta[12]] (delta[3])'
              d Sin[beta[12]] (beta[11])'   d Sin[beta[12]] (delta[2])'
JACR196[5] -> --------------------------- + --------------------------- +
                          2                             2
     d Sin[beta[12]] (delta[3])'
>    ---------------------------
                2
              -(d (beta[12])')
JACR196[6] -> ----------------
                     2
JACR199[4] -> -(JACR147[5] Sin[beta[10]] (beta[9])') -
>     JACR148[5] Sin[beta[10]] (delta[1])' -
>     JACR149[5] Sin[beta[10]] (delta[3])'
              d Sin[beta[10]] (beta[10])'
JACR199[5] -> --------------------------- +
                          2
>     (-(Cos[beta[10]] JACR148[6]) + JACR148[4] Sin[beta[10]])
>      (delta[1])' + (-(Cos[beta[10]] JACR149[6]) +
>        JACR149[4] Sin[beta[10]]) (delta[3])'
JACR199[6] -> Cos[beta[10]] JACR147[5] (beta[9])' +
>     Cos[beta[10]] JACR148[5] (delta[1])' +
>     Cos[beta[10]] JACR149[5] (delta[3])'
JACR200[4] -> -(JACR147[5] Sin[beta[10]] (beta[9])') -
>     JACR148[5] Sin[beta[10]] (delta[1])' -
>     JACR148[5] Sin[beta[10]] (delta[3])'
              d Sin[beta[10]] (beta[10])'
JACR200[5] -> --------------------------- +
                          2
>     (-(Cos[beta[10]] JACR148[6]) + JACR148[4] Sin[beta[10]])
>      (delta[1])' + (-(Cos[beta[10]] JACR148[6]) +
>        JACR148[4] Sin[beta[10]]) (delta[3])'
JACR200[6] -> Cos[beta[10]] JACR147[5] (beta[9])' +
>     Cos[beta[10]] JACR148[5] (delta[1])' +
>     Cos[beta[10]] JACR148[5] (delta[3])'
JACR201[4] -> -(JACR147[5] Sin[beta[10]] (beta[9])') -
>     JACR147[5] Sin[beta[10]] (delta[1])' -
>     JACR147[5] Sin[beta[10]] (delta[3])'
              d Sin[beta[10]] (beta[10])'
JACR201[5] -> --------------------------
                          2
JACR201[6] -> Cos[beta[10]] JACR147[5] (beta[9])' +
>     Cos[beta[10]] JACR147[5] (delta[1])' +
>     Cos[beta[10]] JACR147[5] (delta[3])'
JACR202[1] -> -(Sin[beta[10]] (beta[9])') - Sin[beta[10]] (delta[1])' -
>     Sin[beta[10]] (delta[3])'
JACR202[3] -> Cos[beta[10]] (beta[9])' + Cos[beta[10]] (delta[1])' +
>     Cos[beta[10]] (delta[3])'
              d Sin[beta[10]] (beta[9])'   d Sin[beta[10]] (delta[1])'
JACR202[5] -> --------------------------- + --------------------------- +
                          2                             2
     d Sin[beta[10]] (delta[3])'
>    ---------------------------
                2
              -(d (beta[10])')
JACR202[6] -> ----------------
                     2
JACR205[5] -> (-(Cos[delta[5]] JACR152[6]) + JACR152[4] Sin[delta[5]])
>     (delta[4])'
JACR206[1] -> -(Sin[delta[5]] (delta[4])')
JACR206[3] -> Cos[delta[5]] (delta[4])'
JACR163[1] -> -(JACR161[1] (phi[11])')
JACR163[2] -> -(JACR162[2] (phi[10])') - JACR161[2] (phi[11])'
JACR163[3] -> -(JACR162[3] (phi[10])') - JACR161[3] (phi[11])'
JACR163[4] -> -(JACR162[4] (phi[10])') - JACR161[4] (phi[11])' -
>     JACR160[4] (phi[12])'
JACR163[5] -> -(JACR161[5] (phi[11])') - JACR160[5] (phi[12])'
JACR163[6] -> -(JACR162[6] (phi[10])') - JACR161[6] (phi[11])' -
>     JACR160[6] (phi[12])'
JACR169[1] -> JACR168[1] (beta[2])'
JACR169[3] -> JACR168[3] (beta[2])'
JACR169[4] -> JACR167[4] (beta[1])'
JACR169[5] -> JACR167[5] (beta[1])' + JACR168[5] (beta[2])'
JACR169[6] -> JACR167[6] (beta[1])' + JACR168[6] (beta[2])'
JACR173[1] -> JACR172[1] (beta[4])'
JACR173[3] -> JACR172[3] (beta[4])'
JACR173[4] -> JACR171[4] (beta[3])'
JACR173[5] -> JACR171[5] (beta[3])' + JACR172[5] (beta[4])'
JACR173[6] -> JACR171[6] (beta[3])' + JACR172[6] (beta[4])'
JACR177[1] -> JACR176[1] (beta[6])'
JACR177[3] -> JACR176[3] (beta[6])'
JACR177[4] -> JACR175[4] (beta[5])'
JACR177[5] -> JACR175[5] (beta[5])' + JACR176[5] (beta[6])'
JACR177[6] -> JACR175[6] (beta[5])' + JACR176[6] (beta[6])'
JACR180[5] -> JACR179[5] (delta[3])'
JACR184[5] -> JACR182[5] (delta[3])'
JACR184[6] -> JACR182[6] (delta[3])'
JACR188[5] -> JACR186[5] (delta[3])'
JACR188[6] -> JACR186[6] (delta[3])'
JACR191[5] -> JACR190[5] (delta[4])'
JACR197[1] -> JACR196[1] (beta[12])'
JACR197[3] -> JACR196[3] (beta[12])'
JACR197[4] -> JACR195[4] (beta[11])' + JACR194[4] (delta[2])' +
>     JACR193[4] (delta[3])'
JACR197[5] -> JACR195[5] (beta[11])' + JACR196[5] (beta[12])' +
>     JACR194[5] (delta[2])' + JACR193[5] (delta[3])'
JACR197[6] -> JACR195[6] (beta[11])' + JACR196[6] (beta[12])' +
>     JACR194[6] (delta[2])' + JACR193[6] (delta[3])'
JACR203[1] -> JACR202[1] (beta[10])'
JACR203[3] -> JACR202[3] (beta[10])'
JACR203[4] -> JACR201[4] (beta[9])' + JACR200[4] (delta[1])' +
>     JACR199[4] (delta[3])'
JACR203[5] -> JACR201[5] (beta[9])' + JACR202[5] (beta[10])' +
>     JACR200[5] (delta[1])' + JACR199[5] (delta[3])'
JACR203[6] -> JACR201[6] (beta[9])' + JACR202[6] (beta[10])' +
>     JACR200[6] (delta[1])' + JACR199[6] (delta[3])'
JACR207[1] -> JACR206[1] (delta[5])'
```

```
JACR207[3] -> JACR206[3] (delta[5])''
JACR207[5] -> JACR205[5] (delta[4])''
alpha2[1] -> 0
alpha2[2] -> 0
alpha2[3] -> 0
alpha3[1] -> 0
alpha3[2] -> 0
alpha3[3] -> 0
alpha4[1] -> 0
alpha4[2] -> 0
alpha4[3] -> 0
alpha5[1] -> JACR163[1] - (phi[10])'' - Sin[phi[11]] (phi[12])''
alpha5[2] -> JACR163[2] - Cos[phi[10]] (phi[11])'' -
>    ADH90[2, 3] (phi[12])''
alpha5[3] -> JACR163[3] - Sin[phi[10]] (phi[11])'' -
>    ADH90[3, 3] (phi[12])''
alpha6[1] -> 0
alpha6[2] -> 0
alpha6[3] -> 0
alpha7[1] -> JACR169[1] + Cos[beta[2]] (beta[1])''
alpha7[2] -> (beta[2])''
alpha7[3] -> JACR169[3] + Sin[beta[2]] (beta[1])''
alpha8[1] -> JACR173[1] + Cos[beta[4]] (beta[3])''
alpha8[2] -> (beta[4])''
alpha8[3] -> JACR173[3] + Sin[beta[4]] (beta[3])''
alpha9[1] -> JACR177[1] + Cos[beta[6]] (beta[5])''
alpha9[2] -> (beta[6])''
alpha9[3] -> JACR177[3] + Sin[beta[6]] (beta[5])''
alphai1[1] -> (delta[3])''
alphai1[2] -> 0
alphai1[3] -> 0
alphai0[1] -> (delta[1])'' + (delta[3])''
alphai0[2] -> 0
alphai0[3] -> 0
alphai2[1] -> (delta[2])'' + (delta[3])''
alphai2[2] -> 0
alphai2[3] -> 0
alphai3[1] -> (delta[4])''
alphai3[2] -> 0
alphai3[3] -> 0
alphai4[1] -> JACR197[1] + Cos[beta[12]] (beta[11])'' +
>    Cos[beta[12]] (delta[2])'' + Cos[beta[12]] (delta[3])''
alphai4[2] -> (beta[12])''
alphai4[3] -> JACR197[3] + Sin[beta[12]] (beta[11])'' +
>    Sin[beta[12]] (delta[2])'' + Sin[beta[12]] (delta[3])''
alphai5[1] -> JACR203[1] + Cos[beta[10]] (beta[9])'' +
>    Cos[beta[10]] (delta[1])'' + Cos[beta[10]] (delta[3])''
alphai5[2] -> (beta[10])''
alphai5[3] -> JACR203[3] + Sin[beta[10]] (beta[9])'' +
>    Sin[beta[10]] (delta[1])'' + Sin[beta[10]] (delta[3])''
alphai6[1] -> JACR207[1] + Cos[delta[5]] (delta[4])''
alphai6[2] -> (delta[5])''
alphai6[3] -> JACR207[3] + Sin[delta[5]] (delta[4])''
a2[1] -> 0
a2[2] -> 0
a2[3] -> (s[1])''
a3[1] -> 0
a3[2] -> 0
a3[3] -> (s[2])''
a4[1] -> 0
a4[2] -> 0
a4[3] -> (s[3])''
a5[1] -> JACR163[4] - JACR119[4] (phi[11])'' - JACR120[4] (phi[12])''
                      a2 (phi[10])''
a5[2] -> JACR163[5] + -------------- - JACR120[5] (phi[12])''
                            2
a5[3] -> JACR163[6]
a6[1] -> 0
a6[2] -> 0
a6[3] -> (s[4])''
                      ai (beta[2])''
a7[1] -> JACR169[4] + -------------- + ADH224[1, 3] (s[1])''
                            2
a7[2] -> JACR169[5] + JACR123[5] (beta[1])'' + Sin[beta[1]] (s[1])''
a7[3] -> JACR169[6] + ADH224[3, 3] (s[1])''
                      ai (beta[4])''
a8[1] -> JACR173[4] + -------------- + ADH227[1, 3] (s[2])''
                            2
a8[2] -> JACR173[5] + JACR126[5] (beta[3])'' + Sin[beta[3]] (s[2])''
a8[3] -> JACR173[6] + ADH227[3, 3] (s[2])''
                      ai (beta[6])''
a9[1] -> JACR177[4] + -------------- + ADH230[1, 3] (s[3])''
                            2
a9[2] -> JACR177[5] + JACR129[5] (beta[5])'' + Sin[beta[5]] (s[3])''
a9[3] -> JACR177[6] + ADH230[3, 3] (s[3])''
ai1[1] -> 0
ai1[2] -> JACR180[5] + Sin[delta[3]] (s[4])''
ai1[3] -> h2 (delta[3])'' + Cos[delta[3]] (s[4])''
ai0[1] -> 0
ai0[2] -> JACR184[5] + ADH151[3, 4] (delta[3])'' + ADH233[2, 3] (s[4])''
ai0[3] -> JACR184[6] - ADH151[2, 4] (delta[3])'' + ADH233[3, 3] (s[4])''
ai2[1] -> 0
ai2[2] -> JACR188[5] + ADH109[3, 4] (delta[3])'' + ADH234[2, 3] (s[4])''
ai2[3] -> JACR188[6] - ADH109[2, 4] (delta[3])'' + ADH234[3, 3] (s[4])''
ai3[1] -> 0
ai3[2] -> JACR191[5] + Sin[delta[4]] (s[4])''
          -(f (delta[4])'')
ai3[3] -> ----------------- + Cos[delta[4]] (s[4])''
                2
                     d (beta[12])''
ai4[1] -> JACR197[4] + -------------- + JACR143[4] (delta[2])'' +
                            2
>    JACR144[4] (delta[3])'' + ADH240[1, 3] (s[4])''
ai4[2] -> JACR197[5] + JACR142[5] (beta[11])'' +
>    JACR143[5] (delta[2])'' + JACR144[5] (delta[3])'' +
>    ADH240[2, 3] (s[4])''
ai4[3] -> JACR197[6] + JACR143[6] (delta[2])'' +
>    JACR144[6] (delta[3])'' + ADH240[3, 3] (s[4])''
                      d (beta[10])''
ai5[1] -> JACR203[4] + -------------- + JACR148[4] (delta[1])'' +
                            2
>    JACR149[4] (delta[3])'' + ADH247[1, 3] (s[4])''
ai5[2] -> JACR203[5] + JACR147[5] (beta[9])'' + JACR148[5] (delta[1])'' +
>    JACR149[5] (delta[3])'' + ADH247[2, 3] (s[4])''
ai5[3] -> JACR203[6] + JACR148[6] (delta[1])'' +
>    JACR149[6] (delta[3])'' + ADH247[3, 3] (s[4])''
ai6[1] -> JACR152[4] (delta[4])'' + ADH250[1, 3] (s[4])''
ai6[2] -> JACR207[5] + Sin[delta[4]] (s[4])''
ai6[3] -> JACR152[6] (delta[4])'' + ADH250[3, 3] (s[4])''
```